CrossTalk

June 2006 **The Journal of Defense Software Engineering** Vol. 19 No. 6

Why Projects Fail

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JUN 2006**	2. REPORT TYPE	3. DATES COVERED **00-00-2006 to 00-00-2006**
4. TITLE AND SUBTITLE **CrossTalk: The Journal of Defense Software Engineering. Volume 19, Number 6, June 2006**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **OO-ALC/MASE,6022 Fir Ave,Hill AFB,UT,84056-5820**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**32**	

Why Projects Fail

Software Engineering Technology

Open Forum

ON THE COVER

Cover Design by
Kent Bingham

Additional art services
provided by Janna Jensen
jensendesigns@aol.com

Departments

CrossTalk

76 SMXG Co-Sponsor	*Kevin Stamey*
309 SMXG Co-Sponsor	*Randy Hill*
402 SMXG Co-Sponsor	*Diane Suchan*
DHS Co-Sponsor	*Joe Jarzombek*
NAVAIR Co-Sponsor	*Jeff Schwalb*
Publisher	*Brent Baxter*
Associate Publisher	*Elizabeth Starrett*
Managing Editor	*Kase Johnstun*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Nicole Kentta*
Phone	(801) 775-5555
E-mail	crosstalk.staff@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the U.S. Air Force (USAF), the U.S. Department of Homeland Security (DHS), and the U.S. Navy (USN). USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG), Ogden-ALC 309 SMXG, and Warner Robins-ALC 402 SMXG. DHS co-sponsor: National Cyber Security Division of the Office of Infrastructure Protection. USN co-sponsor: Naval Air Systems Command.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 15.

517 SMXS/MDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, or the STSC. All product names referenced in this issue are trademarks of their companies.

Coming Events: Please submit conferences, seminars, symposiums, etc. that are of interest to our readers at least 90 days before registration. Mail or e-mail announcements to us.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Why Do Projects Fail?

Don't we all wish that there were a simple answer to this question? Despite the many opinions, nobody has a straightforward answer that really applies to all projects. The articles in this issue, and many other articles that I reviewed, point to two facts: First, everyone has a theory on why programs fail, and second, there is no singular cause that universally applies. It is, frankly, surprising that the variety of statistics and articles on project failures have little in common. There were, however, a few – just a few – common denominators to project failures. But since I am a glass-is-half-full kind of guy, I would rather view this topic from the side of success. Although I suspect there are more, I only found three common denominators that were consistently mentioned in several articles that I read as key to a project's success, requirements management being the first common denominator. Success was not just defined by well-documented technical requirements, but well-defined programmatic requirements/thresholds. Most articles specifically note requirements creep as a challenge for projects. The driving forces behind requirements creep exist in many forms, and every project should have a systems engineering strategy to manage requirements, which is a good lead to the second common denominator: risk management. Successful programs not only identified risk early on, but specific steps were defined for managing the risk once identified. The third common denominator consistently mentioned was project planning. Without incredible luck, no project can be successful without realistic and thorough up-front planning. To state the obvious, the better the planning, the more likely the outcome will match the plan.

Like most of you, I have my opinions on how to have a successful project. I would like to point you to what I view are some other critical elements of project success. Having served as both an acquirer and supplier of software, I can tell you that a key to success is sound systems engineering. Few projects can be truly successful if they do not take into account the many tentacles that are linked to a project. Suppliers and acquirers must work together to implement good systems engineering. Another element critical to a project's success is careful consideration in the planning phase to the availability of resources. The availability of key people, equipment, facilities, and many other project resources must be taken into account before committing to a schedule. One final element that the 76th Software Maintenance Group (SMXG) has found to be key to a project's success is project management. Managers must adequately manage day-to-day activities, ensure processes are followed, monitor progress, and communicate with the customer.

While I do not mean to imply that these three elements are the only elements of successful projects, I do believe they are among the most critical elements of success. Our track record here at the 76th SMXG is evidence that these elements have certainly contributed to our success. Our 325 software deliveries in the last 24 months have resulted in 100 percent on-time delivery performance. I hope you will take some of these lessons learned as well as the insights from this month's authors and apply them to your project.

We start this month with an article from Capers Jones on the social and technical reasons for project failure, followed by an entertaining article by Alan C. Jost on the importance of communication. Timothy K. Perkins discusses the conclusions of his independent research and how lack of knowledge or the application of that knowledge can lead to project failure. Quentin W. Fleming and Joel M. Koppelman suggest using a simplified version of earned value as a tool for project success. Our recently retired Walt Lipke continues to aid us with his article on statistical methods applied to Earned Value Management. Timothy G. Olson defines short and *usable* (emphasis added) processes. We conclude with Paul Kimmerly, who provides unique insight on how process improvement projects can actually help a project fail rather than help them succeed.

I trust these articles will help in alerting you to warning signs of potential weaknesses that can lead to project failure. I also hope they help you and your team implement successful processes for project success.

Kevin Stamey

Kevin Stamey
Oklahoma City Air Logistics Center, Co-Sponsor

Social and Technical Reasons for Software Project Failures©

Capers Jones
Software Productivity Research, LLC

Major software projects have been troubling business activities for more than 50 years. Of any known business activity, software projects have the highest probability of being cancelled or delayed. Once delivered, these projects display excessive error quantities and low levels of reliability. Both technical and social issues are associated with software project failures. Among the social issues that contribute to project failures are the rejections of accurate estimates and the forcing of projects to adhere to schedules that are essentially impossible. Among the technical issues that contribute to project failures are the lack of modern estimating approaches and the failure to plan for requirements growth during development. However, it is not a law of nature that software projects will run late, be cancelled, or be unreliable after deployment. A careful program of risk analysis and risk abatement can lower the probability of a major software disaster.

Software is an important but troubling technology. Software applications are the driving force of modern business operations, but software is also viewed by many chief executives as one of the major problem areas faced by large corporations [1, 2, 3, 4].

The litany of senior executive complaints against software organizations is lengthy, but can be condensed down to a set of three very critical issues that occur over and over in hundreds of corporations:

1. Software projects are not estimated or planned with acceptable accuracy.
2. Software project status reporting is often wrong and misleading.
3. Software quality and reliability are often unacceptably poor.

When software project managers (PMs) themselves are interviewed, they concur that the three major complaints levied against software projects are real and serious. However, from the point of view of software managers, corporate executives also contribute to software problems [5, 6]. The following are three complaints against top executives:

1. Executives often reject accurate and conservative estimates.
2. Executives apply harmful schedule pressure that damages quality.
3. Executives add major new requirements in mid-development.

Corporate executives and software managers have somewhat divergent views as to why software problems are so prevalent. Both corporate executives and software managers see the same issues, but these issues look quite different to each group. Let us examine the root causes of the five software risk factors:

1. Root causes of inaccurate estimating and schedule planning.
2. Root causes of incorrect and optimistic status reporting.
3. Root causes of unrealistic schedule pressures.
4. Root causes of new and changing requirements during development.
5. Root causes of inadequate quality control.

> *"One advantage that function points bring to early estimation is that they are derived directly from the requirements and show the current status of requirements completeness."*

These five risk areas are all so critical that they must be controlled if large projects are likely to have a good chance of a successful outcome.

Root Causes of Inaccurate Estimating and Schedule Planning

Since both corporate executives and software managers find estimating to be an area of high risk, what are the factors triggering software cost estimating problems? From analysis and discussions of estimating issues with several hundred managers and executives in more than 75 companies between 1995 and 2006, the following were found to be the major root causes of cost estimating problems:

1. Formal estimates are demanded before requirements are fully defined.
2. Historical data is seldom available for calibration of estimates.
3. New requirements are added, but the original estimate cannot be changed.
4. Modern estimating tools are not always utilized on major software projects.
5. Conservative estimates may be overruled and replaced by aggressive estimates.

The first of these estimating issues – *formal estimates are demanded before requirements are fully defined* – is an endemic problem which has troubled the software community for more than 50 years [7, 8]. The problem of early estimation does not have a perfect solution as of 2006, but there are some approaches that can reduce the risks to acceptable levels.

Several commercial software cost estimation tools have early estimation modes which can assist managers in sizing a project prior to full requirements, and then in estimating development staffing needs, resources, schedules, costs, risk factors, and quality [9]. For very early estimates, risk analysis is a key task.

These early estimates have confidence levels that initially will not be very high. As information becomes available and requirements are defined, the estimates will improve in accuracy, and the confidence levels will also improve. But make no mistake, software cost estimates performed prior to the full understanding of requirements are intrinsically difficult. This is why early estimates should include contingencies for requirements changes and other downstream cost items.

The second estimating issue – *historical data is seldom available for calibration of estimates* – is strongly related to the first issue. Companies that lack historical information on staffs, schedules, resources, costs, and quality levels from similar projects are always at risk when it comes to software cost estimation. A good software measurement program pays handsome dividends over time [10].

For those organizations that lack internal historical data, it is possible to acquire external benchmark information from a number of consulting organizations. However, the volume of external benchmark data varies among industries, as do the supply sources.

One advantage that function points bring to early estimation is that they are derived directly from the requirements and show the current status of requirement completeness [11]. As new features are added, the function point total will go up accordingly. Indeed, even if features are removed or shifted to a subsequent release, the function point metric can handle this situation well [12, 13].

The third estimating issue – *new requirements are added but the original estimate cannot be changed* – is that of new and changing requirements without the option to change the original estimate. It is now known that the rate at which software requirements change runs between 1 percent and 3 percent per calendar month during the design and coding stages. Thus, for a project of 1,000 function points and an average 2 percent per month *creep* during design and coding, new features surfacing during design and coding will add about 12 percent to the final size of the application. This kind of information can and should be used to refine software cost estimates by including contingency costs for anticipated requirements creep [14].

When requirements change, it is possible for some projects in some companies to revise the estimate to match the new set of requirements. This is as it should be. However, many projects are forced to attempt to accommodate new requirements without any added time or additional funds. I have been an expert witness in several lawsuits where software vendors were directed by the clients to keep to contractual schedules and costs even though the clients added many new requirements in mid-development.

The rate of requirements creep will be reduced if technologies such as joint application design (JAD), prototyping, and requirements inspections are utilized. Here too, commercial estimating tools can adjust their estimates in response to the technologies that are planned for the project.

The fourth estimating problem – *modern estimating tools are not always utilized on major software projects* – is the failure to use state-of-the-art software cost estimating methods. It is inappropriate to use rough manual *rules of thumb* for important projects. If the costs are likely to top $500,000 and the schedules take more than 12 calendar months, then formal estimates are much safer.

> ***"Several commercial software cost estimation tools have early estimation modes that can assist managers in sizing the projects prior to full requirements, and then in estimating development staffing needs, resources, schedules, costs, quality, and risk factors."***

Some of the commercial software cost estimating tools used in 2006 include: COCOMO II, Construx Estimate, COSTAR, CostXpert, KNOWLEDGE-PLAN, PRICE-S, SEER, SLIM, and SOFTCOST.

For large software projects in excess of 1,000 function points, any of these commercial software cost estimating tools can usually excel manual estimates in terms of accuracy, completeness, and the ability to deal with tricky situations such as staffing buildups and growth rate in requirements.

Estimating tools have one other major advantage: when new features are added or requirements change, redoing an estimate to accommodate the new data usually only takes a few minutes. In addition, these tools will track the history of changes made during development and, hence, provide a useful audit trail.

The fifth and last of the major estimating issues – *conservative estimates may be overruled and replaced by aggressive estimates* – is the rejection of conservative or accurate cost estimates and development schedules by clients or top executives. The conservative estimates are replaced by more aggressive estimates that are based on business needs rather than on the capabilities of the team to deliver. For some government projects, schedules may be mandated by Congress or by some outside authority. There is no easy solution for such cases.

The best solution for preventing the arbitrary replacement of accurate estimates is evaluating historical data from similar projects. While estimates themselves might be challenged, it is much less likely that historical data will be overruled.

It is interesting that high-tech industries are usually somewhat more sophisticated in the use of estimating and planning tools than financial services organizations, insurance companies, and general manufacturing and service groups. The high-tech industries such as defense contractors, computer manufacturers, and telecommunication manufacturers need accurate cost estimates for their hardware products, so they usually have estimating departments that are fully equipped with estimating tools that also use formal estimating methods [15].

Banks, insurance companies, and *low-technology* service companies do not have a long history of needing accurate cost estimates for hardware products so they have a tendency to estimate using informal methods and also have a shortage of estimating tools available for software PMs.

Root Causes of Incorrect and Optimistic Status Reporting

One of the most common sources of friction between corporate executives and software managers is the social issue that software project status reports are not accurate or believable. In case after case, monthly status reports are optimistic that all is on schedule and under control until shortly before the planned delivery when it is suddenly revealed that everything was not under control and another six months may be needed.

What has long been troubling about software project status reporting is the fact that this key activity is severely underreported in software management literature. It is also undersupported in terms of available tools and methods.

The situation of ambiguous and inadequate status reporting was common even in the days of the *waterfall model* of software development. Inaccurate reporting is even more common in the modern era where the *spiral model* and other alternatives such as *agile methods* and the object-oriented paradigm are supplanting traditional methods. The reason is that these non-linear software development methods do not have the same precision in completing milestones as did the older linear software methodologies.

The root cause of inaccurate status reporting is that PMs are simply not trained to carry out this important activity. Surprisingly, neither universities nor many in-house management training programs deal with status reporting.

If a project is truly under control and on schedule, then the status reporting exercise will not be particularly time consuming. Perhaps it will take five to 20 minutes of work on the part of each component or department manager, and perhaps an hour to consolidate all the reports.

But if a project is drifting out of control, then the status reports will feature *red flag* or warning sections that include the nature of the problem and the plan to bring the project back under control. Here, more time will be needed, but this is time very well spent. The basic rule of software status reporting can be summarized in one phrase: No surprises!

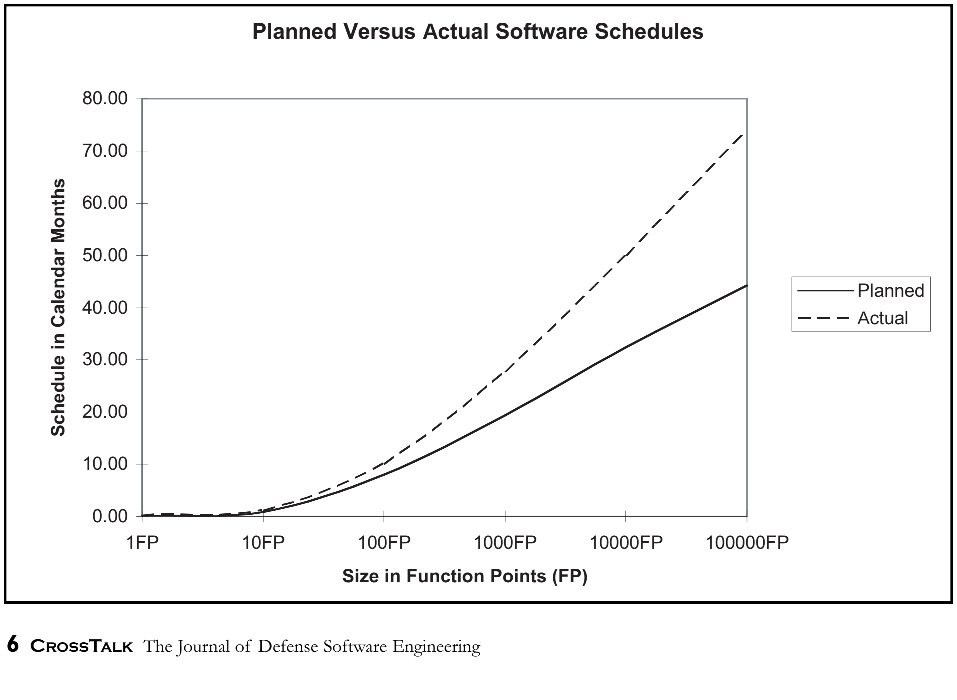
The monthly status reports should consist of both quantitative data on topics such as current size and numbers of defects and also qualitative data on topics such as problems encountered. Seven general kinds of information are reported in monthly status reports:

1. Cost variances (quantitative).
2. Schedule variances (quantitative).
3. Size variances (quantitative).
4. Defect removal variances (quantitative).
5. Defect variances (quantitative).
6. Milestone completions (quantitative and qualitative).
7. Problems encountered (quantitative and qualitative).

Six of these seven reporting elements are largely quantitative, although there may also be explanations for why the variances occur and their significance.

The most common reason for schedule slippage, cost overrun, and outright cancellation of a major system is that they contain too many bugs or defects to operate successfully. Therefore, a vital element of monthly status reporting is recording data on the actual number of bugs found compared to the anticipated number of bugs. Needless to say, this implies the existence of formal defect and quality estimation tools and methods.

Figure 1: *Planned Versus Actual Schedules for Software Projects*



Not every software project needs the rigor of formal monthly status reporting. The following kinds of software need monthly status reports:

- Projects whose total development costs are significant (>$1,000,000).
- Projects whose total development schedule will exceed 12 calendar months.
- Projects with significant strategic value to the enterprise.
- Projects where the risk of slippage may be hazardous (such as defense projects).
- Projects with significant interest for top corporate management.
- Projects created under contract with penalties for non-performance.
- Projects whose delivery date has been published or is important to the enterprise.

The time and effort devoted to careful status reporting is one of the best software investments a company can make. This should not be a surprise: status reports have long been used for monitoring and controlling the construction of other kinds of complex engineering projects.

During the past 20 years, a number of organizations and development approaches have included improved status reporting as a basic skill for PMs. Some of these include the Project Management Institute, the Software Engineering Institute's (SEI) Capability Maturity Model® (CMM®), the reports associated with the Six Sigma quality methodology, and the kinds of data reported when utilizing International Organization for Standardization (ISO) Standards.

Unfortunately, from examining the status reports of a number of projects that ended up in court for breach of contract, inaccurate status reporting still remains a major contributing factor to cost overruns, schedule overruns, and also to litigation if the project is being performed under contract.

Root Causes of Unrealistic Schedule Pressures

Unrealistic schedule pressure by executives or clients is a common software risk factor. There are four root causes for unrealistic schedule pressure:

1. Large software projects usually

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

have long schedules of more than 36 months.

2. PMs are not able to successfully defend conservative estimates.
3. Historical data from similar projects is not available.
4. Some kind of external business deadline affects the schedule.

Figure 1 shows U.S. industry experiences derived from several thousand software projects. The upper curve shows the average delivery time in calendar months, while the lower curve shows the planned or desired delivery time. The larger the project, the greater the gap between the actual delivery date and the planned delivery date of the application [16].

Executives may be arbitrary in their decisions, but they are seldom stupid. While corporate executives might want a large software project finished in 24 months, they will almost certainly accept a 36-month schedule as a fact of life (if they know that not one of 50 similar projects within their industry has ever been completed in less than 36 months). Estimates might be overruled, but accurate historical data will probably keep schedule pressure from becoming unrealistic.

The most difficult problem to solve is when some kind of external business deadline affects the project schedule. Unfortunately, these business deadlines are usually outside the control of either PMs or technical personnel. Examples of external business deadlines include contractual obligations, the starting dates of new laws that require software support, or some kind of technical situation such as those associated with the Y2K problem.

Such external fixed dates cannot be changed, or at least not changed by project personnel. Therefore a combination of cutting back on functions, plus staff overtime, remains the most common method for dealing with fixed and unchanging delivery dates. If the mandated schedule is quite impossible to achieve, then a more drastic option would be project cancellation.

Root Causes of New and Changing Requirements During Development

The root causes of requirements changes are dynamic businesses. Real-world requirements for software must change in response to new business needs. However, average change rates of 2 percent per calendar month indicate that the methods used for gathering and analyzing the initial requirements are inadequate and should be improved.

By counting function points from the original requirements and then counting again at the time of delivery, it has been found that the average rate of requirements growth is about 2 percent per calendar month from the nominal completion of the requirements phase through the design and coding phases.

The total accumulated volume of new or changing requirements can top 50 percent of the initial requirements when function point totals at the requirements phase are compared to function point totals at deployment. The state-of-the-art requirements change control includes the following:

- A joint client/development change control board.
- Use of JAD to minimize downstream changes.
- Use of formal prototypes to minimize downstream changes.
- Formal review of all change requests.
- Revised cost and schedule estimates for all changes under 50 function points.
- Prioritization of change requests in terms of business impact.
- Formal assignment of change requests to specific releases.
- Use of automated change control tools with cross-reference capabilities.

> *“More than 50 years of empirical studies have proven that projects with effective quality control cost less and have shorter schedules than similar projects with poor quality control.”*

One of the observed byproducts of the usage of formal JAD sessions is a reduction in downstream requirements changes. Rather than having unplanned requirements surface at a rate of 1 percent to 3 percent every month, studies of JAD by IBM and other companies have indicated that unplanned requirements changes often drop below 1 percent per month due to the effectiveness of the JAD technique.

Prototypes are also helpful in reducing the rates of downstream requirements changes. Normally, key screens, inputs, and outputs are prototyped so users have some *hands-on* experience with an example of the completed application.

However, changes will always occur for large systems. It is not possible to freeze the requirements of any real-world application.Therefore, leading companies are ready and able to deal with changes and do not let them become impediments to progress; some form of iterative development is a logical necessity.

Root Causes of Inadequate Quality Control

Effective software quality control is the most important single factor that separates successful projects from delays and disasters. The reason for this success is that finding and fixing bugs is the most expensive cost element for large systems, and it takes more time than any other activity.

The root cause for poor quality control is lack of solid empirical data on the cost effectiveness of a good quality control program. More than 50 years of empirical studies have proven that projects with effective quality control cost less and have shorter schedules than similar projects with poor quality control. However, a distressing number of PMs are not aware of the economics of quality control [5, 10].

Successful quality control involves defect prevention, defect removal, and defect measurement activities. The phrase *defect prevention* includes all activities that minimize the probability of creating an error or defect in the first place. Examples of defect prevention activities include the use of the Six Sigma approach, the use of JAD for gathering requirements, the use of formal design methods, the use of structured coding techniques, and the use of libraries of proven reusable material.

The phrase *defect removal* includes all activities that can find errors or defects in any kind of deliverable. Examples of defect removal activities include requirements inspections, design inspections, document inspections, code inspections, and many kinds of testing [17, 18].

The phrase *defect measurement* includes measures of defects found during development and also defects reported by customers after release. These two key measures allow leading companies to calculate their defect removal efficiency rates, or the percentages of defects found prior to release of software applications. Supplemental measures such as severity levels, code complexity, and defect repair rates are also useful and important. Statistical analysis of defect origins and root-cause analysis are beneficial, along with the key measurements of cost and defect repairs [10].

Some activities benefit both defect prevention and defect removal simultaneously. For example, participation in design and code inspection is very effective in terms of defect removal and also benefits defect prevention. Defect prevention is aided because inspection participants learn to avoid the kinds of errors that inspections detect.

Successful quality control activities include defect prevention, defect removal, and defect measurements. The combination of defect prevention and defect removal activities leads to some very significant differences in the overall numbers of software defects between successful and unsuccessful projects.

For projects in the 10,000 function point range, the successful ones accumulate development totals of around 3.0 defects per function point and remove about 96 percent of them before customer delivery. In other words, the number of delivered defects is about 0.12 defects per function point or 1,200 total latent defects. Of these, about 10 percent – or 120 – would be fairly serious defects. The rest would be minor or cosmetic defects.

By contrast, the unsuccessful projects accumulate development totals of around 7.0 defects per function point and remove only about 85 percent of them before delivery. The number of delivered defects is about 1.05 defects per function point or 10,500 total latent defects. Of these, about 15 percent – or 1,575 – would be fairly serious defects. This large number of serious latent defects after delivery is very troubling for users. If a project has more than about 7.0 defects per function point and less than 85 percent removal efficiency, it will probably be cancelled because it can never successfully exit testing, and the test cycle will be hopelessly protracted.

One of the reasons why successful projects have such a high defect removal efficiency compared to unsuccessful projects is the use of design and code inspections [17, 18]. Formal design and code inspections average about 65 percent efficient in finding defects. They also improve testing efficiency by providing better source material for constructing test cases.

Unsuccessful projects typically omit design and code inspections and depend purely on testing. The omission of up-front inspections causes three serious problems:

1. The large number of defects still present when testing begins slows the project to a standstill.
2. The *bad fixes*[1] injection rate for projects without inspections is alarmingly high.
3. The overall defect removal efficiency associated with only testing is not sufficient to achieve defect removal rates higher than about 80 percent.

> ***"The most common reason for schedule slippages, cost overruns, and outright cancellation of major systems is that they contain too many bugs or defects to operate successfully."***

Fortunately, the SEI, ISO quality standards, and the Six Sigma approach have benefited quality control activities throughout the past 20 years. As a result, an increasing number of large projects have been successful compared to similar projects done in the 1980s.

However, for very large projects above 10,000 function points in size, missed delivery dates, cost overruns, and outright terminations remain distressingly high even in 2006. The industry is improving, but much more improvement is needed.

Summary and Conclusions

Large software projects are very hazardous business ventures. For projects above 10,000 function points, cancellations, delays, and cost overruns have been the norm rather than the exception.

Careful analysis of the root causes of large software project delays and disasters indicate that most of the problems stem from inaccurate estimation, inaccurate status reporting, lack of historical data from similar projects, and suboptimal quality control.

All of these root causes can be minimized or even eliminated by the adoption of formal estimating methods and tools, formal monthly status reports of both quantitative and qualitative data, collecting historical data, and improving quality control methods. Large software projects will never be without risk, but if the risks can be brought down to acceptable levels, both clients and corporate executives will be pleased.◆

Note

1. The term *bad fixes* refers to secondary defects accidentally injected by means of a patch or defect repair that is itself flawed. The industry average is about 7 percent, but for unsuccessful projects the number of bad fixes can approach 20 percent; i.e. one out of every five defect repairs introduced fresh defects [14]. Successful projects, on the other hand, can have bad-fix injection rates of only 2 percent or less.

References

1. Yourdon, Ed. Death March – The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects. Upper Saddle River, NJ: Prentice Hall, 1997.
2. Glass, R.L. Software Runaways: Lessons Learned from Massive Software Project Failures. Prentice Hall, 1998.
3. Johnson, James. "The Chaos Report." West Yarmouth, MA: The Standish Group, 2000.
4. Ewusi-Mensah, Kweku. Software Development Failures. Cambridge, MA: Massachusetts Institute of Technology Press, 2003.
5. Jones, Capers. Assessment and Control of Software Risks. Prentice Hall PTR, 1994.
6. Jones, Capers. Patterns of Software System Failure and Success. Boston, MA: International Thomson Computer Press, 1995.
7. Boehm, Barry. Software Engineering Economics. Englewood Cliffs, NJ: Prentice Hall, 1981.
8. Jones, Capers. "Sizing Up Software." Scientific American Magazine Dec. 1998: 104-111.
9. Jones, Capers. Estimating Software

Costs. New York, NY: McGraw Hill, 1998.

10. Kan, Stephen H. Metrics and Models in Software Quality Engineering. 2nd ed. Boston, MA: Addison-Wesley Professional, 2002.
11. Jones, Capers. Applied Software Measurement. 2nd ed. New York, NY: McGraw Hill, 1996.
12. Garmus, D. and D. Herron. Function Point Analysis – Measurement Practices for Successful Software Projects. Boston, MA: Addison-Wesley Professional, 2001.
13. International Function Point Users Group (IFPUG). IT Measurement – Practical Advice from the Experts. Boston, MA: Addison-Wesley, 2002.
14. Jones, Capers. Software Quality – Analysis and Guidelines for Success. Boston, MA: International Thomson Computer Press, 1997.
15. Jones, Capers. Software Assessments, Benchmarks, and Best Practices. Boston, MA: Addison-Wesley Professional, 2000.
16. Jones, Capers. Conflict and Litigation Between Software Clients and Developers. Narragansett, R.I.: Software Productivity Research LLC, 2005.
17. Radice, Ronald A. High Quality, Low Cost Software Inspections. Andover, MA: Paradoxicon Publishing, 2002.
18. Wiegers, Karl E. Peer Reviews in Software – A Practical Guide. Boston, MA: Addison Wesley Professional, 2002.

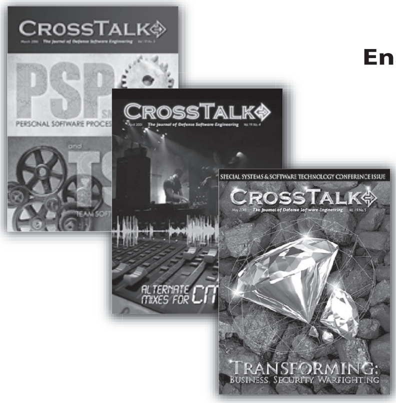
About the Author

Capers Jones is currently the chairman of Capers Jones & Associates, LLC. He is also the founder and former chairman of Software Productivity Research, LLC (SPR), where he holds the title of Chief Scientist Emeritus. He is a well-known author and international public speaker, and has authored the books "Patterns of Software Systems Failure and Success," "Applied Software Measurement," "Software Quality: Analysis and Guidelines for Success," "Software Cost Estimation," and "Software Assessments, Benchmarks, and Best Practices." Jones and his colleagues from SPR have collected historical data from more than 600 corporations and more than 30 government organizations. This historical data is a key resource for judging the effectiveness of software process improvement methods. The total volume of projects studied now exceeds 12,000.

Software Productivity
Research, LLC
Phone: (877) 570-5459
(973) 273-5829
Fax: (781) 273-5176
E-mail: cjones@spr.com

CALL FOR ARTICLES

If your experience or research has produced information that could be useful to others, CROSSTALK can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for three areas of emphasis we are looking for:


CrossTalk
PSP
Personal Software Process
CrossTalk
Alternate Mixes for
CrossTalk
Transforming: Business Security Warfighting

Enabling Technologies for Net-Centricity
January 2007
Submission Deadline: August 21

Agile Development
February 2007
Submission Deadline: September 18

COTS Integration
March 2007
Submission Deadline: October 16

Please follow the Author Guidelines for CROSSTALK, available on the Internet at <www.stsc.hill.af.mil/crosstalk>. We accept article submissions on all software-related topics at any time, along with Letters to the Editor and BackTalk.

COMING EVENTS

July 5-7
18th International Conference on Software Engineering and Knowledge
San Francisco, CA
www.ksi.edu/seke/seke06.html

July 6-8
SEDE 2006
15th International Conference on Software Engineering and Data Engineering
Los Angeles, CA
www.isp.mu-luebeck.de/sede06/index.htm

July 16-19
WMSCI 2006
The 10th World Multi-Conference on Systemics, Cybernetics, and Informatics
Orlando, FL
www.iiisci.org/wmsci2006/website/default.asp

July 16-19
SERP 2006
The 3rd Symposium on Risk Management and Cyber-Informatics
Orlando, FL
www.iiisci.org/rmci2006/website/default.asp

July 23-28
Agile 2006
Minneapolis, MN
www.agile2006.com

July 24-28
Practical Software and Systems Measurement (PSM) 10th Annual Users' Group Conference
Vail, CO
www.psmsc.com/Events.asp

April 16-19, 2007
2007 Systems and Software Technology Conference

STC Systems & Software Technology Conference

www.sstc-online.org

What We've Got Here Is ... Failure to Communicate

Alan C. Jost
Raytheon Company

The tagline from Cool Hand Luke *(1967)* [1] *has often been modified from its original. The Captain (Strother Martin) tells the recalcitrant chain gang prisoner Luke (Paul Newman): "What we've got here is ... failure to communicate," not "What we have here is a failure to communicate." We do not even quote the quote correctly. This article is a look at situations where communication among team members is a critical factor in the potential failure of a program or its success. This is not a deep technical article, but I believe it is thought-provoking. When humans communicate in written, verbal, and nonverbal forms, many times the receiver misses the intended meaning. The "failure to communicate" is the root cause for many program failures more times than we would admit or appreciate.*

Throughout my career, I have experienced a number of *program failures* (even the term failure is relative and subject to a wide range of meanings depending on the individuals participating in the discussion). These program failures can be directly related back to the basic tenet of this article: *failure to communicate*.

Everyone reading this more than likely has had a similar experience and could add to the following situations. This article is not written as an indictment against any one individual, organization, or program; it is written as a lighthearted look at how things that seem so simple can become major stumbling blocks because of our *failure to communicate*. But I do not want to just dwell on the failures, so a couple of good examples of how participants were able to communicate are also presented. Each of the situations is generalized by using groups as examples.

NASA Mars Probe [2]

One of the most dramatic failures of a project caused by failure to communicate was the NASA probe project in 1999. The probe, the Mars Climate Orbiter, was to orbit Mars to gather climatic data. The Orbiter, at a cost of about $125 million, traveled more than 400,000,000 miles to get to the planet. Upon arrival, the Orbiter entered an orbit 60 miles too low, and since it was not built to withstand the Mars atmosphere, was destroyed. The design calculations used to place the spacecraft into orbit were made in imperial measures in terms of *pounds force*. The software team, however, developed the *burn* control software using metric measurements and units in terms of *newtons*. While the error was less than 0.000015 percent, it was enough to be fatal to the mission. The communication error was only uncovered during the post-mortem of the failed mission. This was a major failure to communicate between teams of intelligent, experienced professionals who did not check even the most obvious items in the design and implementation of the probe. "What we've got here is … failure to communicate."

Radar Red Time

In one situation in which I was personally involved, three organizations – the contractor, customer, and operational user – were collaborating to build a large radar system. The new radar was located near the old radar it was replacing. The old radar would not be decommissioned until the new radar was successfully operationally tested. In order to do this, maintenance *red time* of the old radar had to be scheduled when the new radar would be tested; this is where the three organizations *failed to communicate*. Through many planning meetings for red time, each group had a different interpretation of what exactly red time was. The meetings were productive and provided for a detailed operational test schedule. However, each organization had a different interpretation of the red time that created the resultant operational test schedule. The failure to communicate between the organizations was discovered at the first operational test event when the contractor requested that the old radar be turned off.

> "One of the most dramatic failures of a project caused by a failure to communicate was the NASA probe project in 1999."

The contractor assumed that red time meant the old radar would be turned off so they could test the new radar without interference from radiation being transmitted from the old radar. The customer assumed that red time meant that the old radar, while not turned off, would be placed in a maintenance state where the transmission of radiation would be rerouted through the wave-guides, eliminating a large portion of the ambient radiation. The operational user's version of red time meant that only the transmission lines for the radar data would be disconnected, so a false target would not be transmitted. Well, the reaction from the operational user was, "Turn the radar off!? The radar has never been turned off, and we don't even know how to turn it off, and even worse, we don't know how to turn it back on!" "What we've got here is … failure to communicate."

At the heart of the situation was the klystron, the large tube that generated the radiation used to transmit the radar signal. Once turned on, it had not been turned off for years and there were no procedures to turn it off and back on again. In near real time, the three groups had to communicate with the klystron manufacturer to generate a procedure to minimize the energy and redirect the lower energy down the wave-guides. The new procedure did work, and the power down sequence was successfully repeated numerous times to support the operational testing of the new radar. *Failure to communicate* the concept of red time among the participating organizations could have lead directly to a major schedule impact on the program. It forced real-time communications between the participating organizations and manufacturer, resulting in the power down procedure. If the power down procedure failed, it would also have caused a major impact to the program. The procedure worked and the major schedule impact was avoided. What we've got here is … communication!

Contract Negotiations

In another example, we have the customer and contractor negotiating the functionality included in the contractor's proposal. During the negotiations, it was mentioned by the customer that they only had two-thirds of the proposed price in their budget. The contractor was requested to reduce their bid to match the customer's budget and to eliminate the functionality needed to hit the target reduced-proposal price. The proposal team developed the new proposal with reduced functionality to meet the customer's budget and provided the updated information to the negotiating team.

Somehow, some way, the reduced functionality was not accurately communicated to the customer. "What we've got here is … failure to communicate."

It came to light at the first customer contractor system specification review when the software technical lead presented the reduced functionality list. The reaction from the customer was not anticipated. Where were the missing functions? The ones that were eliminated to reduce the bid were the functions they were asking about and the wheels started to fall off. Under the contract, the contractor had to develop the functionality directed by the customer, whether in the specification or not, and the contractor would have to recoup the costs through the country's court system. Eventually, the program resulted in delivery of the system with the full functionality, which the customer assumed they were going to get for the reduced price that matched their budget. The extra functionality, however, required the contractor to fund the additional work. In the end, the court sided with the contractor, and the customer ended up paying for the full functionality by reimbursing the contractor for the additional funding. While eventually remedied, the initial *failure to communicate* made the entire program a contentious affair between customer and contractor.

Communication Systems vs. Communications

These three situations indicate the importance of eliminating the *failure to communicate* among program team members. It is not that we do not have adequate communication systems to communicate with, we have an overabundance of communication and collaborative systems: telephones, cell phones, walkie-talkies, blueberries, blackberries, e-mail, v-mail, fax, eRooms, Docushare, meeting rooms, Sametime (Lotus instant messaging and Web conferencing), and a multitude of other communication and collaboration systems. This is not the problem. The problem is the clear transmission of ideas and concepts between program team members that is at the heart of the problem. "What we've got here is … failure to communicate."

As the reader, you probably have examples of programs where the communication among team members was very good and the project turned out to be a success. To see the impact of good communication leading to successful projects, I like to look to the television show *The Apprentice*. The projects on the show are contrived to be completed in a short period of time to fit the presentation of the project in a one-hour time slot. It is interesting to see that almost 100 percent of the time the team that had good communication with their customer-judge, focus groups, and/or among the team members had the successful project.

> *"It was an overwhelming victory for the project manager who communicated with his potential customers. What we've got here is … communication!"*

The Apprentice – Mural [3]

In one project, two teams had to develop an advertising mural for a new electronic game. The murals were to be done in Harlem. One team was led by a project manager who came from a neighborhood similar to Harlem and she *knew* what would be a good advertising mural. Since she *knew* what the *customers* would like, she knew how to create the mural to attract customers to buy the electronic game. The other team was led by a project manager who came from an upper-edge society and was kind of a geek; he immediately set out to get feedback from a customer focus group in the neighborhood where the mural would be placed. He wanted to find out what was important to them as far as electronic games were concerned. He not only talked with the kids who would use the games, but with the parents who would ultimately purchase the games. Well, guess which team won the project? Which mural did a better job in selling the product? Was it the know-it-all from the 'hood, or was it the geek who communicated with the people in the neighborhood focus group? It was an overwhelming victory for the project manager who communicated with his potential customers. What we've got here is … communication!

The Apprentice – Solstice [4]

A second *The Apprentice* project that demonstrated the importance of communication was the development of a sales brochure to describe the new Pontiac Solstice Roadster. One team was led by and consisted of all men who naturally knew exactly what it would take to sell the new two-seat, convertible roadster. The other team was led by a woman, who, by her own admission, was not much into cars. The male-led team took the approach of making the car a macho-type of machine that would attract good-looking women to the car's male driver, while the female-led team spoke with the General Motors representatives about how they wanted the car to be portrayed. Well, you do not have to be a wizard to guess who won this project management contest. The female-led team won because the project manager captured what the executives communicated they wanted in the sales brochure. Even more importantly was that the Pontiac executives, who were also the judges of the two brochures, decided to use the brochure designed by the female-led team as the actual Solstice brochure in Pontiac showrooms across the nation. While the projects are somewhat contrived to support the premise behind the show, they do demonstrate that the ability to communicate is critical to the success of the project – any project involving a team of people attempting to accomplish a task.

Apollo 13 – Recovery [5]

The original Apollo 13 problem was caused when the number two oxygen tank in the service module exploded because of a short circuit in the oxygen tank that occurred during a routine *stirring* procedure. This problem was not the result of a *failure to communicate*. What I am using this dramatic mission failure example for is to demonstrate the success achieved with the ability of the NASA Apollo ground team to communicate effectively, not only between themselves to develop solutions, but also to communicate those solutions to the Apollo 13 crew. The initial explosion also caused the number one oxygen tank to fail and the fuel cells that supplied the command module with electricity to have problems. In the initial 90 minutes, it was brainstormed by the ground crew to use the Lunar Lander as a lifeboat for the crew. However, the Lunar Lander was designed to be used for 45 hours only, and the return mission around the moon would take 90 hours. There was plenty of oxygen with barely enough electrical power to make the return journey. The foreseeable problem was the eventual build up of carbon dioxide in the spacecrafts. There were enough lithium hydroxide canisters in the command module and Lunar Lander between them, but the command module square canisters were not compatible with the round openings in the Lunar Lander module control system. The Houston mission control team gave the brainstorming team the materials available only to the Apollo 13 crew. The brainstorming team had to come up with the solution to the Apollo 13 *square-peg-in-a-round-hole* problem. Once they came up with the solution, they had to communicate that solution to the crew to implement. Using plastic bags, tape, cardboard, and the square canisters themselves, the brainstorming team came up with the solution. They were able to communicate the solution to the crew in time for their implementation, and the rest is history. What we've got here is … communication.

Summary

Human-to-human communication is critical in managing programs. This is even recognized in the Capability Maturity Model® Integration where stakeholder involvement, reviews with higher levels of management, and other process areas (specific and generic practices) are based on *not failing to communicate*.

I hear you was one of the most popular phrases in the late '90s. It generally translated as one person understood what the other person meant to say. While the words truly mean that you physically heard the words spoken, a more appropriate response would have been *I understood you*. I leave you with just two famous quotes. The first is a small, simple example of a failure to communicate, and the second is an excellent example of precise communication.

In the movie *Apollo 13*, astronaut Jim Lovell (Tom Hanks) tells Mission Control: "Houston, we have a problem." The line has often been misquoted as "Houston, we've got a problem." The historical quote from Apollo 13's Commander Jim Lovell was: "Houston, we've *had* a problem." The actual historic exchange was the following (the times are in mission times in hours, minutes, and sections after launch) [5]:

- 55:55:20 – Swigert: "Okay, Houston, we've had a problem here."
- 55:55:28 – Lousma: "This is Houston. Say again please."
- 55:55:35 – Lovell: "Houston, we've had a problem. We've had a main B bus undervolt."

By now, it is readily apparent the importance of communication. So in conclusion, an example of precise communication is appropriate. Again, a bit contrived, but it makes the point. In the movie *The Fugitive* during the scene right after the train wreck where Dr. Richard Kimball (Harrison Ford) escapes, U.S. Marshal Samuel Gerard (Tommy Lee Jones) has to take over a just-formed, very large search team of local police who are extremely reluctant to be led by the Wyatt Earp-type marshal. He communicates precisely what he needs done. In one short, memorable speech he states his requirements:

> Listen up, ladies and gentleman. Our fugitive has been on the run for 90 minutes. Average foot speed over uneven ground, barring injury, is four miles an hour. That gives us a radius of six miles.
>
> What I want out of each and every one of you is a hard target search of every gas station, residence, warehouse, farmhouse, henhouse, outhouse, and dog house in that area. Checkpoints go up in 15 miles. Your fugitive's name is Dr. Richard Kimball. Go get him!" [6]

Any questions on how clear his communication was? In real estate, the most important thing is location, location, location. In program management it is communication, communication, communication.◆

References

1. Cool Hand Luke. Dir. Stuart Rosenberg. Perf. Paul Newman, George Kennedy, J.D. Cannon, Lou Antonio, and Robert Drivas. Warner Brothers: 1967.
2. Nickson, David, and Suzy Siddons. Project Disasters & How to Survive Them. London and Sterling, VA: Kogan Page, 2006.
3. "The Writing On the Wall." The Apprentice NBC. 24 Feb. 2005.
4. "A Lovely Drive." The Apprentice NBC. 14 Apr. 2005.
5. Nickson, David, and Suzy Siddons. Project Disasters & How to Survive Them. London and Sterling, VA: Kogan Page, 2006.
6. The Fugitive. Dir. Andrew Davis. Perf. Harrison Ford and Tommy Lee Jones. Warner Brothers: 1993.

About the Author

Alan Jost (Lt. Col., U.S. Air Force, retired) is a senior software program manager in the Raytheon Northeast Software Engineering Center (SWEC) where he works multiple tasks: Software Engineering Process Group Executive Committee for SWEC's Capability Maturity Model Integration^{SM} Level 5 sustainment, process engineer on the AutoTrac III Air Traffic Control Product Line, and DD(X) ExComms Software Cross Product Team. Jost joined Raytheon in 1991 after 20 years in the Air Force. He has served in a variety of line management, process engineering, and software task management roles in SWEC, and served intermittently as Software Engineering Process Group Chair between 1993 and 2001.

Raytheon
Mail Stop 3-1-3914
1001 Boston Post RD
Marlborough, MA 01752
Phone: (508) 490-4282
Fax: (508) 490-1366
E-mail: alan_c_jost@raytheon.com

Knowledge: The Core Problem of Project Failure

Timothy K. Perkins
Software Technology Support Center

After having participated in more than 10 independent reviews of major acquisition projects, and having been associated with project/program management throughout my military career, I assert that the cause of project failures is knowledge: either managers do not have the necessary knowledge, or they do not properly apply the knowledge they have.

Having led and participated in more than 10 Independent Expert Program Reviews[1] (IEPRs) for the Software Technology Support Center and the Tri-Service Assessment Office, and having spent my military career as a project/program manager, several individuals have asked if there is a common thread among programs or projects that are having difficulty. The answer is yes. Some expect the thread to be project planning, others risk management, and others expect one of the other project management themes. However, the root causes can be reduced to two issues: either project managers do not have the knowledge they need, or they do not properly apply the knowledge they have. Throughout the remainder of this article, I will refer only to projects, but will mean both programs and projects.

Some may consider these two issues to be too simplistic. However, if they take their pet principle from the Capability Maturity Model Integration (CMMI®)[2], the Project Management Institute's Project Management Body of Knowledge (PMBOK)[3], the Tri-Service Assessment[4], Typology, etc., and then ask why a project is having difficulty in that particular area, it still boils down to either a project manager lacks knowledge of the particular principle, or that the knowledge has not been applied properly. The Project Failure Cause-Effect Diagram (Figure 1 and sidebar) shows this relationship, admittedly through a great leap of logic between all the causes leading to project failure (155).

Lack of Knowledge

The first of these primary causes of project failure – project managers do not know what to do (115) – is the easiest to correct. One solution is to provide the necessary training, remedying the problem of managers not receiving necessary training (100). Jack Ferguson, while teaching workshops for the Software Engineering Institute, used to use the phrase "Just too late training." What he meant was that too often, training in a particular topic is provided too far in advance of need. Students often have been heard saying, "Why do I need to learn this stuff? I'll never use it." When training is provided *just too late*, the students have already realized the need for the training and can readily see how the principles being taught can help them be more successful in accomplishing their projects. This does not mean that the project is in trouble prior to receiving training, but that training is provided at the appropriate time in the project life cycle. For example, providing in-depth training on project closeout prior to project initiation has less value than providing such training as the project enters the project closeout phase of the life cycle, and project members now realize the importance of having the training.

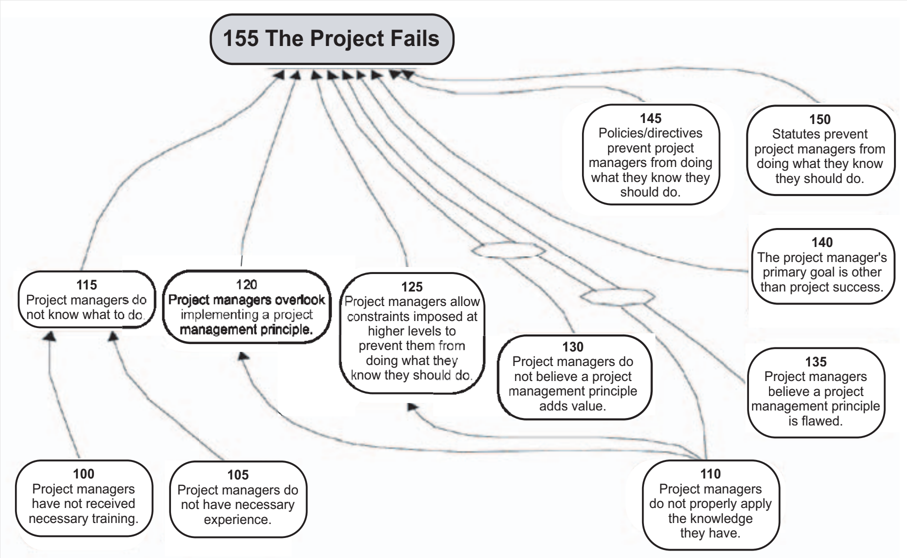
This, however, implies that the project has a plan for training. Many organizations that have undergone IEPRs had neither individual training plans nor an allocated budget to provide necessary training, implying that senior management had either not been trained in the need for establishing an organization training program or had not applied what they had been taught.

The project manager's lack of

The Project Failure Cause-Effect Diagram

The Cause-Effect Diagram is read by locating the entity at the tail of an arrow, and reading it preceded by the word *If*. Then read the entity at the head of the arrow, preceded by the word *then*. For example, the arrow between entities 100 and 115 of the figure would be read: *If* 100 Project managers have not received necessary training, *then* 115 Project managers do not know what to do. If there are several causes joined by an ellipse, read the *If* only once, with other contributing cause statements joined by *and*. For example, the arrows between entities 110 and 155 and 130 and 155 read: *If* 110 Project managers do not properly apply the knowledge they have *and* 130 Project managers do not believe a project management principle adds value *then* 155 The project fails.

Figure 1: *The Project Failure Cause-Effect Diagram*


155 The Project Fails
145 Policies/directives prevent project managers from doing what they know they should do.
150 Statutes prevent project managers from doing what they know they should do.
140 The project manager's primary goal is other than project success.
115 Project managers do not know what to do
120 Project managers overlook implementing a project management principle.
125 Project managers allow constraints imposed at higher levels to prevent them from doing what they know they should do
130 Project managers do not believe a project management principle adds value.
135 Project managers believe a project management principle is flawed.
100 Project managers have not received necessary training.
105 Project managers do not have necessary experience.
110 Project managers do not properly apply the knowledge they have.

[®] CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

experience (105) is often regarded as a *what came first, the chicken or the egg?* problem. An individual needs experience to be a good manager, but how can one gain experience unless they are given the opportunity to manage? The answer is to allow managers to develop experience by learning to manage small projects before being given responsibility for large projects. However, this is not always the case. Some project managers have been put in charge of acquisition category one (ACAT 1) projects after having received only the 14-week Defense Systems Management College program management course, having no prior experience in system acquisition.

Improper Application of Knowledge

Project managers not properly applying the knowledge they have (110) – the second root cause of project failure – is more difficult to remedy. There are five associated causes and effects for this cause shown in Figure 1. If the issue is one of overlooking the implementation of a project management principle (120), i.e., just a lapse of memory in what needed to be done, then gentle reminders from subordinates, peers, or supervisors can be a catalyst to correct the omission. Program offices can provide mentoring to project managers to deal with such oversight. This mentoring can be used to provide refresher training to those who either have forgotten what to do or may not be familiar with current policy, directives, procedures, etc.

Stories abound of projects tasked to accomplish the impossible based on imposed constraints (e.g., cost, resources, performance, etc.) (125). For example, the project manager may determine that a project will take 36 months to complete, but downward direction is to provide *rubber on the ramp* in 30 months. The only way to accomplish this is to take shortcuts, eliminating such things as peer reviews, close configuration management, risk management, etc. Unfortunately, taking shortcuts usually results in lengthening a project because of rework. What would have taken 36 months in a well-planned project now takes 48 months. The only real cure that I know of for this issue is personal integrity: the willingness to tell higher management that their tasking is impossible, and then helping senior management realize that based on current technology, policy, directives, procedures, budget availability, etc., that a project cannot be completed as directed. Note that having a repository of measurements from previous projects or using available industry data can help sway senior management in setting reasonable expectations.

Some project managers consider some mandated project management practices to be of little value (130). For example, some consider preparing a formal project plan to be a waste of effort. They consider a project schedule prepared by using one of the popular project management software packages to be adequate[5]. Others may consider peer reviews of code or documents to use more resources than the value gained. Rather than discuss their concerns with senior management, they choose to ignore the principle with which they disagree. When the belief that a practice does not add value is coupled with the decision to not implement it, project failure can occur. Project managers should discuss their concerns with senior managers and resolve them. One of the best ways to deal with these concerns is to use historical data from other projects to validate the benefit of certain practices or to show the consequences of not following certain practices.

> *"Within the past few years, changes in the Federal Acquisition Regulation and the Department of Defense 5000 series have given managers greater latitude in managing their projects, but the pendulum can always swing back to impose more constraints, whether they are warranted or not."*

Books espousing various project management philosophies and methodologies abound. Current policies and directives have mandated some of the philosophies and methodologies while ignoring others. Some project managers may not believe that the mandated philosophies are the best approaches to use (135). Rather than implement what they consider to be flawed methods, they choose to follow what they consider to be proper principles. During management reviews, it becomes evident that the mandated methods are not being used, and the project manager is directed to implement the mandated method. Resources are now consumed in either getting on the *right* track or in trying to gain approval to not use the mandated methodology – resources that are usually scarce. This use of scarce resources has a negative impact on the chance of project completion within the originally allocated cost and schedule. One solution to this problem is to gain agreement during the project initiation phase of the project life cycle among all parties regarding the project management principles to be used for the remainder of the project.

But what if the application of sound project management principles to ensure project success is not the primary goal of the project manager (140)? While this problem seldom occurs, a few project managers may see their current position as a stepping stone for advancement to higher positions. Their goal may be to show good short-term results at the expense of overall project success. They build a *house of cards* hoping it will stay together and that the collapse will not occur until after the project manager is reassigned. This also is a personal integrity issue. The development of personal integrity in project managers is well beyond the scope of this article.

Though the issues of policy/directives (145) or statutes (laws) (150) preventing a project manager from properly managing a project seldom occur, they both must be recognized as potential causes of project failure. Within the past few years, changes in the Federal Acquisition Regulation and the Department of Defense (DoD) 5000 series have given managers greater latitude in managing their projects, but the pendulum can always swing back to impose more constraints, whether they are warranted or not.

A Word of Caution

If projects continue to have difficulty even after applying what are considered to be sound project management principles, maybe it is the principle that is in error. One definition of idiocy is to continue do things the same way and to expect different results. However, doing things the same way is an underlying premise of process improvement, i.e., the process must be consistent (in statistical control) before it can be improved. If project performance is not as desired, even after consistent application of the project management principle, the underlying principle should be analyzed to determine the reason for the continual shortfall. Perhaps the principle is not as sound as some would have you believe.

Conclusion

Think about known project failures then do a root cause analysis in your mind. For example, why wasn't a project properly planned? Why was risk management not properly implemented? Why was the project not properly tracked? Why were any of the other project management principles not properly followed? I believe that if you evaluate the potential causes, you will reach the same conclusion that I have: it all boils down to knowledge. Either individuals do not have the necessary knowledge, or they have not properly applied the knowledge.◆

Notes

1. IEPRs were called for in Interim Regulation, DoD 5000.2-R, January 1, 2001, paragraph 2.6.8 for ACAT I-III Software Intensive Programs using the words, "The acquisition strategy shall describe the planned use of independent expert reviews for all ACAT I through ACAT III software-intensive programs." The Defense Acquisition Guidebook, Vers. 1.0, (10/17/2004), paragraph 11.14 still encourages the reviews, but the wording has been changed to read, "The program manager for an Acquisition Category ID or IC program that requires software development to achieve the needed capability should convene an independent expert program review after Milestone B and prior to the system Critical Design Review. The program manager, or other acquisition official in the program chain of command up to the component acquisition executive, should also consider independent expert program reviews for Acquisition Category IA, II, and III programs. The independent expert review team should report review findings directly to the program manager." This guidebook is available at <http://akss.dau.mil/dag/>.
2. The CMMI is available for free download at <http://www.sei.cmu.edu/cmmi/models/>.
3. The PMBOK guide is available for purchase at <http://www.pmibookstore.org/PMIBookStore/productDetails.aspx?itemID=358&varID=1>.
4. The Tri-Service Assessment Office is part of the Office of the Under Secretary of Defense (Acquisition, Technology, and Logistics). The Web site is <http://www.acq.osd.mil/tai/>, but is currently under construction.
5. Data Item Description (DID) DI-IPSC-81427A, Software Development Plan, provides a good template of the areas that should be included in a software project plan. The Institute of Electrical and Electronics Engineers Standard 1058-1998, Standard for Software Project Management Plans, is another source for an example project plan. As described in the PMBOK guide and the cited DID, a project plan is more than a schedule.

About the Author

Timothy K. Perkins currently is an independent consultant and has supported the Software Technology Support Center at Hill Air Force Base, Utah in providing consulting services to the U.S. Air Force, U.S. Department of Defense, and other government agencies. Perkins has led and participated in several Independent Expert Program Reviews of Acquisition Category I acquisition programs as well as three Capability Maturity Model® Integration Acquisition Module pilot assessments. He has been involved in software process improvement for the past 17 years. Before retiring from the Air Force, Perkins was Acquisition Professional Development Program Level 3-certified in Project Management and Systems Planning, Research, Development and Engineering, and Level 1 in Test and Evaluation. Perkins is a Project Management Institute-certified Project Management Professional and an authorized Theory of Constraints Thinking Process instructor. Perkins has a Bachelor of Science in Electrical Engineering from Brigham Young University and a Master of Business Administration from the University of Phoenix

E-mail: perkinst8c@aol.com

Start With "Simple" Earned Value On All Your Projects

Quentin W. Fleming and Joel M. Koppelman
Primavera Systems, Inc.

When one hears the term Earned Value Management (EVM), there is a tendency to immediately think of the American National Standards Institute/Electronic Industries Alliance (ANSI/EIA)-748-1998 required by many government agencies and private contractors. We take exception to this formal definition of EVM. While the ANSI/EIA-748 Standard does require full compliance with 32 precise criteria, and will result in EVM, we suggest that a simple form of EVM can be gained by implementing just 10 of these formal criteria on all projects – even software projects.

In 1965, the U.S. Air Force acquisition managers defined 35 criteria which they felt would capture the essence of earned value management (EVM), and also satisfy their need to oversee the work that was being performed for them by private industry. Two years later, the Department of Defense (DoD) adopted these same criteria as their Cost/Schedule Control Systems Criteria (C/SCSC). These 35 standards were then consistently applied to all cost type and incentive type contracts for the next three decades.

Then in 1996, after a rewrite of the 35 C/SCSC criteria by private industry, the DoD accepted the rewriting/rewording of these criteria under a new title called the Earned Value Management System (EVMS). The total number of criteria was reduced to 32. Gone were the incomprehensible terms of Budgeted Cost for Work Schedule, Budgeted Cost for Work Performed, and Actual Cost of Work Performed, etc. In their place were titles like planned value, earned value, and actual costs. People (even busy executives) could understand the concept without the need for special training or a translator being present.

Private industry in the form of the National Defense Industrial Association (NDIA) took the defined criteria concept one step further. In June 1998, the NDIA obtained acceptance of the EVMS in the form of the American National Standards Institute, termed the ANSI/EIA-748 Standard. The good news in this story is that there has been a consistent application of the earned value criteria concept applied for more than 40 years. The earned value criteria have met the test of time.

The bad news is that these criteria were originally written for applications to complex major system acquisitions. Further bad news is that the original 35 criteria – and the reworded 32 criteria – can be overly prescriptive to most of the projects in the world, in our opinion. They are great for major systems, but likely too much for most projects. Somehow a way must be found to capture the important fundamentals of earned value without overly prescribing requirements, which often discourages individuals wanting to adopt a technique to better manage their projects. And, as the ANSI/EIA-748 Standard becomes more commonplace, likely taking the form of a Federal Acquisition Requirements clause issued in routine procurements, a way must be found to scale back the full requirements to meet the needs of most projects – even small software projects.

Since 1996, the authors have been advocating a *simple* form of EVM for all projects, not just major complex systems [1]. Their intent is not to take issue with the full application of all 32 criteria whenever the project risks and complexities warrant full application. However, we think all projects could benefit from "*A methodology for … objectively measuring project performance and progress* [2][1]." EVM, the fundamental principles, should be applied to any project, of any size, in any industry.

We have studied the formal criteria concept and have summarized just 10 fundamental steps which are necessary to implement a *simple* (low-end) form of EVM on any project. Perhaps it might be called earned value light or earned value for the masses.

There are 10 minimum requirements necessary to employ *simple* earned value. This is a good place to start the process. It will set the foundation for employing EVM, which can be easily expanded to satisfy all 32 criteria, should that be desired sometime in the future.

Each of the following 10 fundamental steps will also make reference to a specific EVMS (ANSI/EIA-748 Standard) criterion.

Step 1: You must define the scope (objectives and deliverables) of the project. Satisfying this first criterion is where we lose many projects, but it is critical to the earned value method. Certain types of projects, notably software, often give up at this point and refuse to go further. *Too difficult* is the cry. Management often relinquishes, and the project defaults to simply comparing their cost expenditures – planned costs versus actual costs. What a shame.

On any project, you must define the work to be done if for no better reason than to know where you are at all times and when you are done. To the extent that you can based on past experience, you must define 100 percent of the scope of the project. This is true for any project, but it is particularly critical on any project in which you intend to measure earned value performance.

With earned value, we constantly focus on the authorized work that has been completed plus management's official authorized budget for the completed work. We express the status as being 18 percent complete, 27 percent complete, 55 percent complete, and so forth. Point: If we have not defined what constitutes 100 percent of the project, how can we ever measure our point of percentage completion? We can not.

How does one define a new job when specific details are often lacking? There are no absolute answers. But one of the most useful of all tools available to any project manager is the Work Breakdown Structure (WBS). The WBS is to the project manager what the organization chart is to the executive. A WBS allows the project manager to define a new endeavor by laying out all the assumed work within the framework of the WBS and then decomposing each element into measurable work packages.

Additionally, once the WBS is assumed to constitute a reasonable portrayal of the new project, the WBS can then be used to take the next critical steps in the project planning process, including make-or-buy analysis, risk assessment, scheduling, estimating, and ultimately the authorization of budgets to proceed.

(Reference: EVM Criterion No. 1 [2]: *Define the authorized work elements for the program. A WBS tailored for effective internal management control is commonly used in this process.)*

Step 2: You must determine who will perform the defined work, including the identification of all critical procurements. It is important to a project to decide who will perform the defined work. Experienced workers generally work better and faster than inexperienced people, but they also cost more. Often, using an experienced work force is a good investment. However, sometimes the project's own organization may have no experience in developing a critical new technology, and the project must procure the effort from another company. These choices are called *make* or *buy* decisions, and selecting those items that must be purchased for the project is an essential extension of the scope definition process.

Why is it important to identify the work which must be procured outside? Because project procurements (versus in-house work) create *non-forgiving* legal arrangements. Formal contracts must be executed. If you commit to buy something that is not what you need, or the requirements must be changed, such changes will be accommodated, but at a price. Sellers love to have changes in scope. Each change gives them an opportunity to *get well* from a competitive bid. The earlier the procured work is identified and responsibilities assigned, the better such packages can be managed by the project.

By contrast, internal budgets can be executed in a more informal way, and the fact that everyone is on the same payroll allows for some margin of slack. But there is no slack with the procured work. Procurements must be done properly at the start or the project will pay a price.

Lastly, whether the project work is done by the project's own organization or procured from outside the company, the measurement and reporting of progress must take place. Inside or outside, the project must be able to continuously measure the earned value versus the actual cost of the work being performed.

(Reference: EVM Criterion No. 2: Identify the program organizational structure including the major subcontractors responsible for accomplishing the authorized work, and define the organizational elements in which work will be planned and controlled.)

Step 3: You must plan and schedule the defined work. The earned value technique could be thought of as representing nothing more than a good scheduling system, but with authorized resources (the budgets) embedded into the schedule. The schedule reflects the authorized scope and timeframe, and the budget is earned for work as it is accomplished.

A formal scheduling system is thus necessary to the employment of earned value because it is the vehicle that describes the project scope, the planned value, and then measures the resulting earned value. The project schedule is vital to earned value because it reflects the project manager's baseline *planned value* for everyone to follow.

On more complex projects, there should be some method used to isolate the constraints between one task and other tasks. What work is holding up other work? Typically to satisfy this requirement, some form of critical path methodology will need to be employed. The critical path (and near critical paths) on a project must be aggressively managed in conjunction with negative earned value schedule variances.

> *“The good news in this story is that there has been a consistent application of the earned value criteria concept applied for more than 40 years. The earned value criteria have met the test of time.”*

A behind-schedule variance indicates that the project is falling behind its baseline plan. If any late tasks are also on the critical path, or they represent high risk tasks, they must be aggressively managed to successful completion.

(Reference: EVM Criterion No. 6: Schedule the authorized work in a manner which describes the sequence of work and identifies the significant task interdependencies required to meet the requirements of the program.)

Step 4: You must estimate the required resources and formally authorize budgets. Once the project scope has been fully defined and subsequently planned and scheduled, the next requirement is to estimate the resource requirements (budgets) for all defined tasks. Some projects follow the start-up sequence of scope, schedule, and budget while others follow scope, budget, and schedule. Software projects, because they are often driven by the availability of limited resources will schedule the project based on available people. Either way can be correct as long as scope definition comes first.

Each defined WBS element must have a resource value estimated to complete all of the specified work, including changes. Management will then assess the requested resources and approve a value in the form of an authorized budget. Individual WBS budgets should never contain contingencies or management reserves. Reserves or contingencies, if they exist, must be isolated and owned by the project manager.

Remember the rule that *planned value* represents two things: the scheduled work, plus the authorized budget. *Earned value* also represents two things: the completed authorized work, and the same authorized budget. Thus, in order to plan and then measure earned value, one needs to schedule all defined tasks along with the authorized budget necessary to complete the tasks.

All authorized budgets must be achievable in order to have a viable project baseline.

(Reference: EVM Criterion No. 9: Establish budgets for authorized work with identification of significant cost elements [labor, material, etc.] as needed for internal management and for control of subcontractors.)

Step 5: You must determine the metrics to convert planned value into earned value. How does one measure the accomplishment of planned value into earned value? One sets up measurable (verifiable) metrics within the baseline schedules to quantify the authorized work, and then measures the completion of the authorized work. Specific milestones or tasks with weighted values are measured as they are physically performed. Remember, earned value project management is nothing more than managing a project with a resource-loaded schedule.

Since earned value was first introduced, various methods have been devised to measure project performance. However, the most respected methods use some type of discrete measurement. Specific milestones representing points in time are assigned values, which when fully completed, the assigned budgeted values are then earned. Also, tasks are assigned values, which can be measured as they are partially completed, at which time some value is assigned to the completed work through the reporting period.

(Reference: EVM Criterion No. 7: Identify physical products, milestones, technical performance goals, or other indicators that will be used to measure progress.)

Step 6: You must form a performance measurement baseline and determine the points of management control referred to as Control Account Plans (CAPs). Earned value requires use of an integrated project baseline. An integrated baseline means that the defined work must include both the baseline schedule and the authorized budget. Integration takes place within each of the specified WBS elements.

Project management must next specify their points of management focus, referred to in earned value as CAPs [2]. CAPS are placed at selected WBS elements and can best be thought of as sub-projects, project teams, or subdivisions of the total project. The sum of the CAPS will constitute the total project baseline. The actual earned value performance measurement will take place within each of the specified CAPs. Total project performance is simply the summation of all the detailed CAPs, which can be placed at any level of the WBS.

On some commercial type contracts, the total project baseline may sometimes include such things as indirect costs and even profits or fees to match the total authorized project commitment. The project baseline must include whatever executive management has authorized the project manager to accomplish.

Most likely, internal company projects typically do not contain indirect costs or profits. Many (perhaps most) internal project baselines will simply represent the sum of the defined CAPs, which are often made up from direct labor hours only. The authorized project baseline constitutes whatever management has decided it should be.

Note: The referenced EVM criterion No. 8 contains a lot of words, most of which are beyond the requirement for simple earned value applications.

(Reference: EVM Criterion No. 8: Establish and maintain a time-phased budget baseline, at the control account level, against which program performance can be measured. Initial budgets established for performance measurement will be based upon either internal management goals or the external customer-negotiated target cost including estimates for authorized but undefined work. Budget for far-term efforts may be held in higher-level accounts until an appropriate time for allocation at the control account level. On government contracts, if an over-target baseline is used for performance measurement reporting purposes, prior notification must be provided to the customer.)

Step 7: You must record all direct costs by project consistently with the authorized baseline budgets, in accordance with the organization's general books of accounts. This criterion simply requires that project managers be informed as to how much money they have spent on their projects – a simple requirement that some organizations find extremely challenging. The reason is many organizations have been functionally oriented for so long that they have lost their ability to focus on individual project performance. It is absolutely essential that direct costs be identified by project as work progresses.

In order to employ earned value on any project, the actual costs must be aligned to the authorized project budgets. Remember the rule that planned value represents the authorized work plus budget, which is then converted into completed work and the same budget to represent the earned value. Earned value must then be relatable to the actual costs in order to determine the cost efficiency factor, called the Cost Performance Index (CPI). The CPI is likely the single most important metric for any project employing earned value.

> *"… whether the project work is done by the project's own organization or procured from outside of the company the measurement and reporting of progress must take place."*

There is a trend in projects employing earned value to measure performance on a weekly basis. We need to understand what this means, and what it does not mean. Weekly EVM means the measurement of internal direct labor hours. On a weekly basis, the company labor tapes will produce a planned value, earned value, and actual hours for internal direct labor hours only. Direct labor dollars, indirect costs, purchased articles, travel, etc., are generally not available on a weekly basis. Weekly performance measurement takes place on the internal direct labor hours only.

(Reference: EVM Criterion No. 16: Record direct costs in a manner consistent with the budgets in a formal system controlled by the general books of account.)

Step 8: You must continuously monitor the earned value performance to determine cost and schedule departures from the baseline plan: both schedule variances (earned value less the planned value) and cost variances (earned value less the actual costs). Projects employing earned value must monitor their cost and schedule results against the authorized baseline for the duration of the project. Management will focus their primary attention on exceptions to the baseline plan, particularly those that are greater than previously defined acceptable tolerances. Earned value is thus a *management by exception* concept.

A negative earned value schedule variance simply means that the value of the work performed does not match the value of the work scheduled, that is, the project is falling behind in its scheduled work plan. Each behind-schedule task should be assessed as to its criticality. If the late tasks are on the critical path, or if the tasks carry a high risk to the project, then efforts must be taken to get the late tasks back on schedule. However, additional project resources should not typically be spent on low-risk tasks or tasks that have positive critical path float.

The single most important aspect of employing earned value is the cost efficiency readings it provides. The difference between the value of work earned, versus the costs incurred to accomplish the work provides the cost efficiency factor. If the project spends more money than it receives in value, this reflects an overrun condition. Overruns are typically non-recoverable. Overruns expressed as a percentage value have been found to deteriorate unless the project takes aggressive actions to mitigate the condition.

Perhaps of greatest benefit, the earned value cost efficiency rate has been found to stabilize from the 20 percent point of a project completion. The cost efficiency factor, CPI, is thus an important metric for any project manager or portfolio executive to monitor.

(Reference: EVM Criterion No. 22: At least on a monthly basis, generate the following information at the control account and other levels as necessary for management control using actual cost data from, or reconcilable with, the accounting system:

1. *Comparison of the amount of planned budget and the amount of budget earned for work accomplished. This comparison provides the schedule variance.*
2. *Comparison of the amount of the budget earned and the actual (applied where appropriate) direct costs for the same work. This comparison*

provides the cost variance.)

Step 9: Using earned value data, you must forecast the final required costs based on actual performance and keep management apprised so they can take corrective actions if necessary. One of the more beneficial aspects of earned value is that it provides the capability to quickly and independently forecast the total funds required to complete a project, commonly referred to as the estimate at completion. Based on actual cost and schedule performance against the baseline plan, a project is able to accurately estimate the total funds it will require to finish the job within a finite range of values.

Often, management or customers will have a preconceived notion of what final costs should be or what they would like them to be. If the earned value statistical forecast of estimated final costs are greater than the *official* project manager's estimate to complete the project, someone needs to reconcile these professional differences of opinion.

Actual performance results on any project, good or bad, are in effect *sunk costs*. Such costs represent what the project has actually achieved in performance. Thus any improvements in performance must come from the future work – tasks that lie ahead of the project's status date. Earned value allows the project manager to accurately quantify the cost and schedule performance achieved to date. And if the results achieved to date are less than that desired by management, the project can exert a more aggressive posture to influence the future work.

Earned value allows the project to accurately quantify the value of its work it has achieved. It also allows the project to quantify the value of the future work in order to stay within the objectives set for the project by management. Likely, the single most respected method to forecast the final cost results is to assume that the project will continue at its established cost efficiency rate, CPI; it will get better or worse within a narrow, finite range.

(Reference: EVM Criterion No. 27: Develop revised estimates of cost at completion based on performance to date, commitment values for material, and estimates of future conditions. Compare this information with the performance measurement baseline to identify variances at completion important to company management and any applicable customer reporting requirements including statements of funding requirements.)

Step 10: You must manage the authorized scope by approving or rejecting all changes, and incorporating approved changes into the project baseline in a timely manner. The project performance measurement baseline, initially put into place at the project start, is only as good as the management of all proposed new changes to the baseline for the duration of the project. Performance baselines quickly become invalid simply by failing to incorporate changes into the approved baseline with the addition or deletion of added work scope.

All new change requests of the project must be quickly addressed, either by approving such changes or by rejecting them. All project managers should have sufficient authority to say *no*.

In order for the initial baseline to remain valid, every change must be controlled. Maintaining an approved baseline can be as challenging as the initial definition of the project scope at the start of the project.

(Reference: EVM Criterion No. 28: Incorporate authorized changes in a timely manner, recording the effects of such changes in budgets and schedules. In the directed effort prior to negotiation of a change, base such revisions on the amount estimated and budgeted to the program organizations.)

Summary

Earned value project management is not a difficult concept to understand or employ. It is certainly not as complicated a process as some have made it to be throughout the years. We have concluded that effective earned value can be achieved by simply applying these 10 steps and can be applied to any project, of any size, in any industry. Earned value is for the masses.

Again, we are not taking issue with the EVMS or ANSI-EIA-748 Standard. What we are suggesting is starting with the implementation of earned value in a limited way, on all projects, by simply taking the 10 simple steps as outlined here.

As you read over these 10 steps, we hope you come to the conclusion that employing earned value project management consists of nothing more than simply following fundamental best project management processes. As was stated nicely by a gentleman from the United Kingdom:

> Whilst you can practice good project management without EVM, you cannot practice EVM effectively without good project management. [3]

We could not have stated it better.◆

Note

1. The numbers as shown relate to the sequence of the listed criteria in both the EVMS and ANSI-EIA-748 Standard.

References

1. Fleming, Quentin W. and Joel M. Koppelman. Earned Value Project Management. 3rd ed. Newtown Square, PA: Project Management Institute, 2005.
2. Project Management Institute. A Guide to the Project Management Body of Knowledge. 3rd ed. Newtown Square, PA: Project Management Institute, 2004.
3. Crowther, Steve. "Best of British: Earned Value Management." British Association for Project Management June 1999:13.

About the Authors

Quentin W. Fleming is a management consultant specializing in earned value. He has been a consultant to the senior staff at Primavera Systems, Inc. since 1993. He, along with Joel M. Koppelman, is co-author of "Earned Value Project Management," published initially in 1996 by the Project Management Institute. The third edition of this book was released in 2005. His personal Web site is <www.quentinf.com>

14001 Howland WY
Tustin, CA 92780
Phone: (714) 731-0304
Fax: (714) 731-0304
E-mail: quentinf@comcast.net

Joel M. Koppelman is the co-founder and Chief Executive Officer (CEO) of Primavera Systems, Inc. He, along with Quentin W. Fleming, is co-author of "Earned Value Project Management," published initially in 1996 by the Project Management Institute. The third edition of this book was released in 2005. His corporate Web site is <www.primavera.com>.

Primavera Systems, Inc.
Three Bala Plaza West
Bala Cynwyd, PA 19004
Phone: (610) 667-8600

Statistical Methods Applied to EVM: The Next Frontier

Walt Lipke
Retired Software Manager

An objective of Earned Value Management is to provide a means for predicting the outcome of a project. Inherently, the outcome is largely determined in the planning, and completion forecasting commonly occurs with analysis of project performance. Having the project plan, management would like to be able to quantify its risk: What is the likelihood for having a successful project with this plan? How much should be allocated to reserves to achieve a high probability of success? If reserves are constrained to maintain a bid price in the competitive range, what is the probability of having a successful outcome? During project execution, management desires to answer this question: Can we state with confidence when the project can be expected to complete and simultaneously describe its projected final cost? The application of statistical methods facilitates answering these questions. This article describes the elements necessary for performing statistical analysis.

The worth of Earned Value Management (EVM) has been demonstrated over the 35 plus years of application to many projects. There is substantive evidence of its positive influence on project outcome results. EVM fosters several good management practices that contribute to successful project performance: organization, accountability, planning, risk assessment, tracking, reporting, controlling, etc. Overarching these elements, in my opinion, is the most significant contribution to the improvement of the state of management practice is that EVM has brought science to the management of projects. Without numbers, scientific management is not possible. Because of EVM, project managers have numbers with a sound basis. Performance of a project has a quantitative description with meaning. And, in turn, the numerical description provides project managers with information useful for guiding and controlling the project. From a relatively simple concept, a quantum leap has been made for the management of projects – *Earned Value Management.*

Several formulas derived from EVM measures are available for predicting the final cost of projects. These cost prediction formulas have been well studied over the last 15 years. From the research, the EVM community has an understanding of project behaviors. We now know how to calculate the most optimistic and predicted outcome for cost. And, we understand that projects perform less efficiently as they progress toward completion. For very large projects, we know from early results the range of likely final cost outcomes. Significant strides have been made in project management from the use of EVM measures; project managers now have available a few research-derived prediction tools.

Is There a Path to Improved Prediction?

In truth, advancement of outcome prediction knowledge for EVM-based projects has remained stagnant for nearly a decade. The prediction findings cited previously were established several years ago and have not been improved upon. Although there is more than 35 years of numerical evidence of project performance for many types of applications (defense, construction, software, etc.) from several countries, this EVM data is not available for research. If we could only get by the unfounded worry that by divulging our data for completed projects we are somehow giving up *sensitive* information that could somehow negatively impact our company. Possibly, the influence of the Sarbanes-Oxley Act [1] may help to overcome this roadblock to advancement of EVM. Let us hope so. The sharing of data will not only lead to improved prediction methods, it will also promote continuing improvements to EVM itself.

In the previous discussion, I have established that a researcher, desiring to test a theory concerning EVM, has only limited data – specifically, his own. Thus, the question becomes: "What advancement can be made knowing the researcher's hypothesis cannot be fully tested and validated because of the inaccessibility of broad-based data?" At this time, many of you will probably say, "Not much." Even with today's situation, we can improve our capability to predict outcomes. Here is my answer to the question: *Apply well-established statistical methods.* Statistical methods are proven calculation techniques by which one can infer project outcomes with confidence. Using these methods, past performance can provide a vision of the future.

Is It Difficult to Do?

Good question. Without a background in statistics, it may be somewhat overwhelming in the beginning. However, with a small amount of training in the applicable areas and some practice with EVM data, proficiency will come. In the absence of statistical tools applicable to EVM, you will need to develop spreadsheets until the commercial EVM tool sources catch up to the market. Creating the spreadsheets will not be difficult for someone adept at it and can likely be accomplished in semi-professional form within a short amount of time (my estimate is two to three weeks).

Our Focus

Before we lose ourselves in the discussion of statistics, the focus of this article needs to be stated. The objective is to provide project managers the ability to answer the following questions:

- What is the likelihood for having a successful project with this plan?
- How much should be allocated in reserves to achieve a high probability of success?
- If reserves are constrained to maintain the bid price in the competitive range, what is the probability of having a successful outcome?
- Can we state with confidence when the project can be expected to complete and simultaneously describe its projected final cost?

Certainly, with the ability to answer these questions, project managers and their superiors can make better informed decisions. By taking the correct management action at the right time, we can expect improvement in the success rate for projects and the avoidance of failure.

Applying Statistics to EVM

To apply statistical methods, a few properties of data are needed before we can address these questions. First, we need to establish that the data can be described by *Normal* distribution. If it can, then

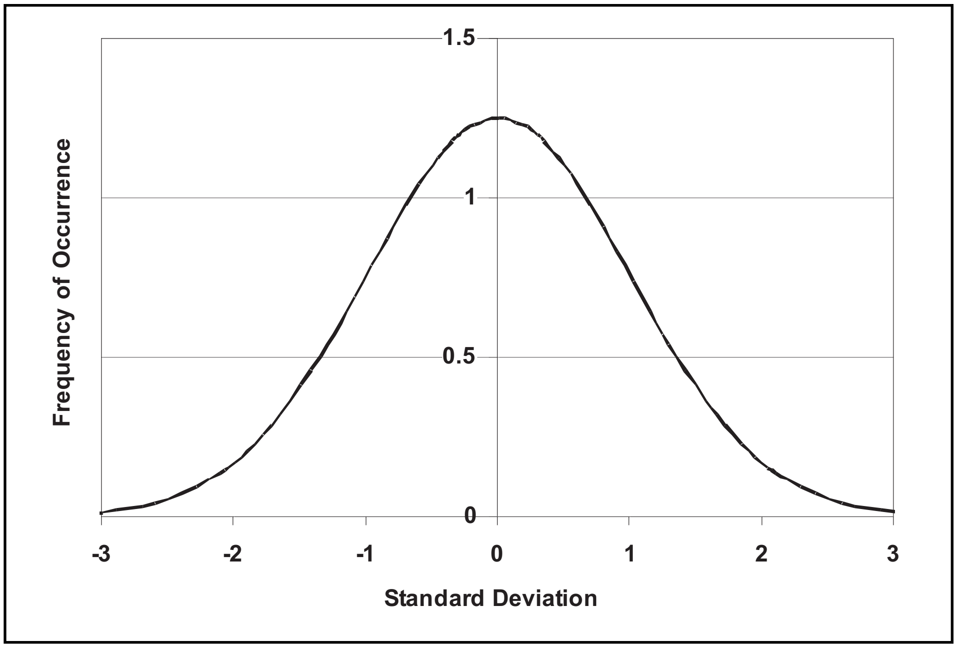


Figure 1: *Normal Distribution*

our ability to draw inferences and make predictions is greatly simplified. The second property is the value representing the mean or average value of the observations. The third property is the variation in the observed data values. These properties are interconnected; without the characterization of the data (i.e., its type of distribution), neither the mean nor the variation can be determined correctly. And without the mean and variation, the focus questions cannot be answered.

Let us assume the observances of the EVM indicator are normally distributed; Figure 1 is an example of Normal distribution. When this is the case, the distribution is symmetrical around its peak, the most frequently observed value. The mean of the distribution is the value associated with the peak. The width or spread of the distribution is a function of the variation in the observed values – the larger the spread, the greater the variation[1].

From this information, inferences or predictions can be made. For example, we can calculate at a specified precision the range of values for the EVM indicator which encompasses its true value or *predicted outcome value*. In statistical terminology, the end values of this range are *confidence limits* (CL). These limits are generally calculated at 90 or 95 percent precision and are commonly termed *xx percent confidence level*. For example, the CL calculated using the 95 percent CL provide a range of values in which we have 95 percent confidence of including the true value of the mean. To make this clearer, I will express it mathematically [2]:

$$CL = Mean \pm Z * \sigma/\sqrt{n}$$

where,

Z is a value representing the 90 or 95 percent confidence level
σ is a number representing the variation in the observed values
n is the number of observances

This equation is not very daunting, and possibly you are beginning to see the usefulness of calculating CL. Clarity with regard to its application should be realized from the coming examples.

Another fundamental needed for having the ability to answer our questions is the calculation of the probability for achieving a specified result. In essence, the calculation obeys the above equation. However, instead of calculating confidence limits, we compute the value of Z [2]:

$$Z = (X - Mean) \div (\sigma/\sqrt{n})$$

where,

X is a value for which an associated probability is desired

From the calculated value of Z, the probability that the true value of the mean is less than or equal to the value X can be obtained from a mathematical table of Normal distribution [2], or by using a spreadsheet function to perform the conversion. For example, the statistical function NORMSINV from Microsoft Excel may be used to perform the calculation.

Although it may not be totally clear at this point, with these two fairly simple equations, every one of the earlier questions can be answered.

Calculation Examples

For understanding, let us perform a few calculations pertinent to our objectives. We will continue with the assumption that the periodic observations of the Cost Performance Index (CPI) are normally distributed. For the example, the cost performance efficiency (cumulative CPI) of a software project is found to be equal to 0.931. The cumulative value of CPI is taken to be a good estimate of the mean of the observations. The variation of the periodic values of CPI, i.e., the estimate of the standard deviation (σ), is equal to 0.340. The number of periodic observations is 16. The level of confidence desired is 90 percent; from a normal distribution table, the value of Z is determined to be 1.645. From this information, we can calculate the confidence limits:

$$\begin{aligned} CL &= Mean \pm Z * \sigma/\sqrt{n} \\ &= 0.931 \pm (1.645) * (0.340 / \sqrt{16}) \\ &= 0.931 \pm 0.140 \\ &= 1.071,\ 0.791 \end{aligned}$$

The values calculated for the confidence limits, 0.791 and 1.071, identify the range for the mean of CPI. Furthermore, we have 90 percent confidence that the true value of the mean of CPI is within these limits.

With this information, we can predict the high and low values of the final cost with 90 percent confidence using the following formula:

$$IEAC = BAC/CL$$

where,

IEAC (Independent Estimate at Completion) is the forecast cost at project completion
BAC (Budget at Completion) is the planned cost for the project

Assuming BAC = $1,000, the range for final cost is $1,264 and $934. Now, assume that in order to not consume all the management reserve, the cost performance efficiency must be greater than or equal to 0.850. Another way of viewing this is the reciprocal of CPI (mean), 1/0.931 = 1.074, must be less than or equal to the reciprocal of 0.850, or 1.176. With these numbers and the parameter values provided in the previous example, the probability of a having a successful project can be computed:

$$\begin{aligned} Z &= (X - \text{Mean}) \div (\sigma/\sqrt{n}) \\ &= (1.176 - 1.074) \div (0.340/\sqrt{16}) \\ &= 0.102 \div 0.085 \\ &= 1.200 \end{aligned}$$

Converting Z (using Normal distribution), we obtain the probability of the project's final cost being less than its allocated budget to be 88.5 percent.

Is It Really That Simple?

No. I wish it was. The previous description of the calculations illustrates the idea in its simplest form, but there are six elements which add complexity:

- Normality.
- Finite population.
- Equal samples.
- Anomalous behavior.
- Fewer than 30 observations.
- Increasing inefficiency.

Recall in the previous discussion and the calculation examples, it was assumed that the periodic values of CPI are normally distributed. This is not the case; the distribution is right-skewed. From previous work, I have shown that by applying logarithms, the distributions of CPI and Schedule Performance Index SPI(t)[2] can be made to appear normal [3]. Figure 2 illustrates the transformation of a right-skewed distribution to its symmetrical normal distribution by the application of logarithms.

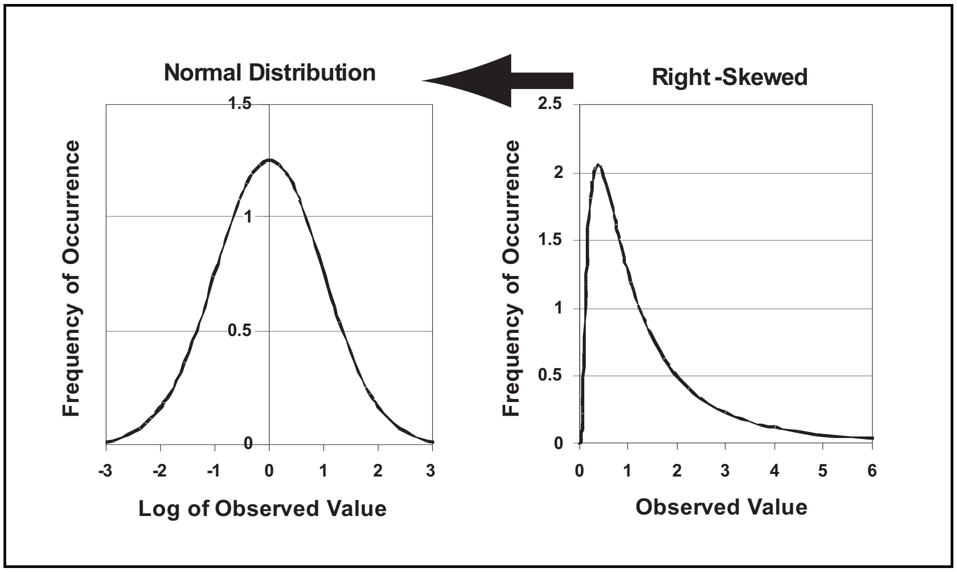
The second element, finite population, is extremely significant. Statistical methods assume the population under examination is infinite. However, projects are finite – they have a start and an end. For finite populations, the statistical calculations must be adjusted. As the project moves toward completion, the adjustment causes the probability of success to move toward 100 percent or zero; i.e., the project completed successfully or it did not. Likewise, the finite population adjustment causes the upper and lower confidence limits to approach each other, concluding at the same value, the mean.

Statistics assumes that each observance is of equal size. For example, if we are trying to infer the proportion of black marbles to white ones in a huge barrel, we might choose to draw independently 10 samples of 10 marbles. It would not be correct statistical practice to draw 10 samples of varying size. In our situation, each observance of CPI represents differing amounts of actual cost. To perform the statistical analysis in the appropriate manner, periodic CPIs must be developed for equal cost samples [4]. From the project data examined to date, the estimate of the variation is slightly smaller for equal cost samples than its value calculated from simply using the reported periodic CPI values.

Certainly, if there is one periodic value that is much different from the remainder, we have to question whether or not to include it in our calculations. By including the anomaly, we might predict a project outcome much differently from the prediction made excluding it. The inclusion of the anomaly has the potential of causing an incorrect management action as well. My recommendation is to identify anomalies using the methods of Statistical Process Control, applying the Shewhart rule only [5]. Removing anomalous behavior improves project outcome prediction and its identification enables appropriate management action.

When the number of observances is fewer than 30, it is accepted practice to perform the statistical calculations using the Student–t distribution[3]. When the number is 30 or greater, Normal distribution is used.

Figure 2: *Transformation to Normal Distribution*



Lastly, from research of CPI behavior, it is known that cost performance efficiency tends to be worse at project completion than it is earlier in the project [6]. Although a similar study of schedule performance behavior has not been made, it is conjectured that SPI(t) behaves analogously to the findings for CPI [7]. Thus, from this tendency to worsen, the forecast final CPI and SPI(t) will generally be less than its respective present value. To account for this behavior, compensation is applied at each of the periods to forecast the final values [7]. The compensation affects the variation calculated; the variation of the compensated periodic values of CPI, or SPI(t), is likely to be somewhat less than for the uncompensated values.

Hopefully these complexities are not an overwhelming deterrent. Obviously, they do add to the calculation burden. However, with some ingenuity all can be handled without much trouble through the use of spreadsheets, dealing with the complexity is really not that difficult. Keep in mind the benefit to your project management having reliable outcome prediction. The value of good prediction far outweighs the discomfort of accommodating the complicating elements discussed.

Calculation Examples – Including Complexity

Let us perform the calculations again and account for the elements adding complexity. For these calculations, assume that none of the observations exhibit anomalous behavior and the distribution is lognormal. Also assume the compensated CPI mean is 0.911 and that the variation of the compensated monthly values is 0.250 for the Normal distribution. Note that both values are somewhat less than those used in the earlier example, just as we would expect. Recall from earlier discussion that the final cumulative CPI tends to be less than the present value, and the variation is smaller from the effects of equal samples and applying compensation. For this example, the total population of observances for the project is 21, and from the previous example the number of observations (n) is equal to 16.

For the confidence limits, the following calculation is made:

$$\ln CL = \ln \text{Mean} \pm Z * \sigma/\sqrt{n} * \text{Adjustment for finite population}^{4}$$

$$
\begin{aligned}
&= \ln (0.911) \pm (1.645) * (0.250/\sqrt{16}) * \\
&\quad \sqrt{((21 - 16)/(21 - 1))} \\
&= -0.093 \pm 1.645 * 0.062 * 0.5 \\
&= -0.093 \pm 0.051 \\
&= -0.042,\ - 0.144 \\
CL &= 0.959,\ 0.866
\end{aligned}
$$

Using the confidence limits, the final cost prediction is calculated: IEAC = $1,155, $1,043.

The probability of having a successful project is computed as follows:

$$
\begin{aligned}
Z &= (\ln X - \ln \text{Mean}) \div \\
&\quad [(\sigma/\sqrt{n}) * \text{Adjustment for finite population}] \\
&= (\ln 1.176 - \ln 1.074) \div [(0.250/\sqrt{16}) * \\
&\quad \sqrt{((21 - 16) / (21 - 1))}] \\
&= (0.163 - 0.093) \div [(0.250/\sqrt{16}) * \\
&\quad \sqrt{((21 - 16)/(21 - 1))}] \\
&= 0.070 \div [0.062 * 0.5] \\
&= 2.240
\end{aligned}
$$

Converting Z, using the Student-t distribution, the probability of having a successful project outcome is determined to be 98.0 percent.

The differences between these estimates and those computed previously are very noticeable. The range of the confidence limits is very much smaller for the more complex calculation ($112 versus $330), thereby causing the final high and low cost estimates to be much closer. The probability of having a successful project is increased by nearly 10 percent for the second calculation (even when using the Student-t distribution). In other words, by accounting for the complexities, the project manager has a much more refined estimate of the final outcome.

Summary

From the past studies performed on EVM measures from large defense contracts, managers and analysts have some ability to forecast the final cost of projects. The ability to advance forecasting beyond its present status, that is to projects which are neither defense related nor large (as are many software or information technology projects), is hampered by the lack of accessible broad-based data for research. Consequently, researchers have little facility to test their hypotheses.

To circumvent the lack of data for experimentation, the application of statistics is proposed. The use of statistical methods for inferring outcomes is a longstanding mathematical approach. The methods applied to EVM measures are shown to be relatively simple in concept. However, several elements are discussed, which cause the application to have added complexity. Including the complexity elements in the method is shown to provide managers with a more refined forecast of project outcome.

Final Remarks

My desire for this article is that it will promote interest in the application of statistical methods to EVM measures. If interest is generated, it is my belief that other positive behaviors may follow:

- Project data records will become more meticulous and, thus, become more useful for further research.
- Data sharing will occur leading to a common EVM data repository for researchers.
- As the use of statistical methods propagates, automated tools will emerge, in turn further expanding the application.
- If this vision of the next *frontier* becomes reality, project management will make another quantum leap forward.◆

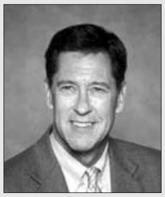
Notes

1. The statistical variation of observed measures is expressed as standard deviations. See [2] (or any text on statistics) for a complete description.
2. SPI(t) is the Schedule Performance Index (time based) and is a measure of schedule performance efficiency [8].
3. The Student-t distribution approaches the Normal distribution as the number of observations becomes large (>30) [9].
4. The adjustment for a finite population is √ [(N–n) / (N–1)], where n is the number of observations made thus far and N is the total population when the project is complete; that is, the number of observations expected to be made.

References

1. The Sarbanes-Oxley Act of 2002. Pub. L. H.R.3763. 7 Jul. 2002. <http://news.findlaw.com/hdocs/docs/gwbush/sarbanesoxley072302.pdf>
2. Crow, E.L., F.A. Davis, and M.W. Maxfield. Statistics Manual. New York: Dover, 1960.
3. Lipke, W. “A Study of the Normality of Earned Value Management Indicators.” The Measurable News Dec. 2002: 1-16.
4. Lipke, W. “Achieving Normality for Cost.” The Measurable News Fall/Winter 2003: 1-11.
5. Pitt, H. SPC for the Rest of Us. Reading, MA: Addison-Wesley, 1995.
6. Christensen, D.S., and S.R. Heise. “Cost Performance Index Stability.” National Contract Management Journal 25 (1993): 7-15.
7. Lipke. W. “Connecting Earned Value to the Schedule.” CROSSTALK June 2005 <http://www.stsc.hill.af.mil/crosstalk/2005/06/index.html>.
8. Lipke, W. “Schedule Is Different.” The Measurable News Summer 2003: 31-34.
9. Wagner, S.F. Introduction to Statistics. New York: Harper Collins, 1992.

About the Author



Walt Lipke recently retired as the deputy chief of the Software Division at the Oklahoma City Air Logistics Center. The division employs approximately 600 people, primarily electronics engineers. He has more than 35 years of experience in the development, maintenance, and management of software for automated testing of avionics. In 1993, with his guidance, the Test Program Set and Industrial Automation (TPS and IA) functions of the division became the first U.S. Air Force activity to achieve Level 2 of the Software Engineering Institute’s Capability Maturity Model® (CMM®). In 1996, these functions became the first software activity in federal service to achieve CMM Level 4 distinction. Under Lipke’s direction, the TPS and IA functions became ISO 9001/TickIT registered in 1998. These same functions were honored in 1999 with the Institute of Electrical and Electronics Engineers’ Computer Society Award for Software Process Achievement. Lipke has published several articles and presented at conferences on the benefits of software process improvement and the application of earned value management and statistical methods to software projects. He is the creator of the technique Earned Schedule©, which extracts schedule information from earned value data. Lipke is a professional engineer with a master’s degree in physics.

1601 Pembroke DR
Norman, OK 73072
Phone: (405) 364-1594
E-mail: waltlipke@cox.net

Defining Short and Usable Processes©

Timothy G. Olson
Quality Improvement Consultants, Inc.

Many processes and procedures are large or difficult to use. The situation becomes even worse when complexity is involved. Putting large or difficult-to-use documentation on a Web site does not usually solve the problem. This article describes best practices for defining short and usable processes and procedures. These best practices have been used at real organizations over the last few years to define short and usable processes and procedures. Measurable results include cutting organizational processes and procedures in half while making them more usable (e.g., reducing 600 pages to 300 pages). The objectives of this article are to describe common problems with process documentation, including some human aspects of using process documents, discuss some best practices for defining short and usable processes and procedures, describe some success stories in real organizations, and provide some lessons learned.

Before we get into describing industry best practices, it is good to understand common problems with process documentation. Table 1 lists a summary of common problems.

How do we address the common problem with process documentation? One place to start is to recognize that not all documentation is used the same way. In this article, the term *process documentation* is used interchangeably with *policies, standards, processes, and procedures*. Table 2 contains a list of the types of process documentation and the ways in which these are used.

Figure 1 [2] identifies the types of process documents and some critical relationships among those documents.

Table 1: *Common Problems Process Documentation*

Problem	Description
1. Too Big	Most process documentation is too big. Blaise Pascal once said, "I have made this letter longer than usual because I lack the time to make it shorter." This quote applies to most processes and procedures. Process documentation should be short, concise, and usable.
2. Not Enough Pictures	Most processes and procedures lack pictures and diagrams. If a picture is worth a thousand words, process documentation should have more pictures. Good process documentation should be a mixture of pictures and words. The best pictures are well-thought-out diagrams. The best diagrams for process documentation are process models.
3. Poorly Designed Documentation	Processes and procedures usually violate documentation design and good writing principles. Principles such as chunking[1] (i.e., seven plus or minus two), consistency, etc., are usually not used [1].
4. Unusable and *One Size Fits All*	Most processes and procedures are not designed with customers and users in mind, making them hard to use. Much documentation also has the *one-size-fits-all* mentality because it does not consider expert, intermediate, and beginner users.
5. Mixed Information Types	Policies, standards, processes, procedures, and training are all different types of information [2]. Most process documentation mixes these different types of information into the same paragraphs as if they were all used the same way. Each one of these document types has a different usage scenario.
6. Written Sequentially	Process documentation is not a novel, and is not meant to be read linearly (i.e. from beginning to end). Process documentation is reference material that is meant to be used non-linearly. This is why labeling is critical so that users can find information quickly [1].
7. Hard-to-Find Information Fast	Users of documentation will look for information for a few minutes. But if they cannot find the information quickly, many times they will give up in frustration and not use the processes or procedures. This can lead to serious nonconformance problems for many organizations.
8. Shelfware	Most process documentation becomes shelfware (i.e. collects dust on a shelf). Online processes (e.g. on an intranet) must be well designed or they will also become unused Web-ware.

Process Documentation Usage Modes

Processes and procedures have different levels of users [3]. Some users have never used the process (i.e., beginner users). Some users have used the process a few times, but need guidance and lessons learned (i.e., intermediate users). Some users have used the process many times and may even be responsible for running the process (i.e., experts). The next sections will describe the three levels of documentation: expert, intermediate, and beginner.

Expert Mode Documentation

Expert mode documentation is short and concise [3]. When a pilot flies an airplane, he or she does not pull out a training manual; they use expert checklists for takeoff and landing. Expert mode documentation is made for experts and does not contain any training material. See Figure 2 (page 26) for an example of expert mode.

Most people want expert mode documentation because it is short. The problem with expert mode documentation is that not everybody is an expert. For example, not everyone can read a checklist for a rocket scientist (sometimes you really need to be a rocket scientist). Putting expert mode documentation in the hands of non-experts can be dangerous.

Why do experts need documentation if they are experts? Because people can forget things. This is why checklists are so powerful. Experts can also leave your organization, taking precious organizational knowledge with them. This is why expert knowledge should be documented.

Intermediate Mode Documentation

Intermediate mode documentation uses the expert mode documentation, but builds and adds to it by providing guidance and lessons learned. For example, guidance is very useful to people who do not have to follow a process or procedure very often. Even experts forget guidance and lessons learned for an annual process or an infrequently used process. Having guidance available to those who need or want it is very useful. Both Figure 2 and Table 3 (page 26) together are examples of intermediate mode.

Typically, guidance and lessons learned are not auditable. Process phases and procedure steps are required and auditable, but the supporting guidance and lessons learned are there for support only. One best practice is to distinguish between required steps and optional guidance. Some organizations have chosen to label guidance and lessons learned with a guidance label.

Beginner Mode Documentation

Beginner mode documentation uses the intermediate mode documentation, but adds training to it. Beginners should feel free to use the training manuals until they become familiar with the process. Beginners should also be mentored as appropriate. Some processes are simple, and some are complex. Complex processes should have formal training and be followed up by mentoring.

Usage Problem and Solutions

How can an organization afford to provide three versions of the same documentation? Someday software could allow this by just setting a documentation mode (expert/intermediate/beginner) and the user could see the appropriate information. A best practice that solves this problem is to define the process in *chunks* at the intermediate level (i.e., one version in intermediate mode). Add in training for the beginner and the expert can grab the appropriate chunks. Another best practice is that short, expert-mode documentation can also be provided for the experts.

What Is a Good Process?

The purpose of this section is to describe the required process elements necessary to define a *good process*. Table 4 (page 27) describes the practical *who, what*, *where*, *when*, *why*, and *how* process questions [2]. Each question is answered by a key process element. By addressing all the process elements on one page, the five W's (who, what, where, when, and why) can be represented in a diagram on one page in expert mode (Figure 2).

Document Type	Usage
Policy	• Used by senior management to set direction in an organization. • States principles that organizations should follow.
Standard	• Specifies the sections of a document, and provides a description of what goes into those sections. • Makes the content of documents repeatable.
Process	• What happens over time to produce a desired result(s). • Should answer the five Ws: who, what, where, when, and why.
Procedure	• How-to or step-by-step information [1]. Example procedures are checklists, forms, and step/action tables. • Implements part of a process.

Table 2: *Types of Process Documentation and Their Uses*

A good process should include the following:

- Address the five W's and answer the key process questions.
- Have both pictures and words (most people prefer pictures, but some people prefer words).
- Be usable and well written.
- Be well chunked and labeled so chunks can be found quickly [1].
- Be short (e.g., a diagram that answers the five W's on one page).

A Best Practice: Process Modeling

A good process should have pictures. It is said that a picture is worth a thousand words. However, not all pictures are good pictures. Some pictures cause confusion, and some pictures are more harmful than helpful. So what is a good picture? Process modeling is a best practice that helps design good diagrams that address the five W's. For a short and usable example, see Figure 2.

What is a process model? A process model *M* models *R* (reality) if *M* answers questions about *R* [4]. A good process model should answer the key process questions (i.e. the five W's in Table 4). A process model is typically represented by diagrams and powerful notations that represent roles, activities, work products, and the relationships between them.

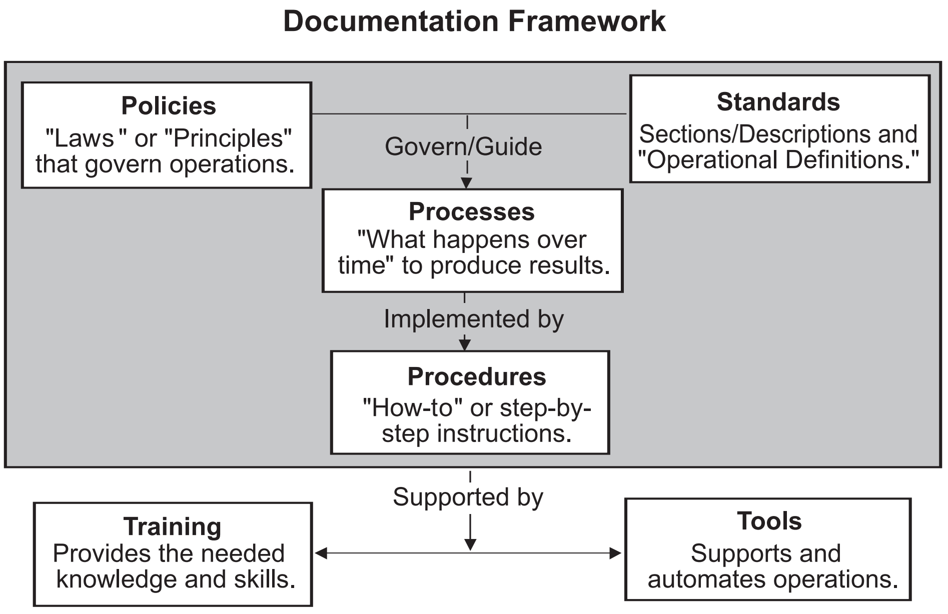
What Is a Good Procedure?

A good procedure consists of *how-to*, *step-by-step* information, and comes in three forms: checklists, forms, and step/action tables [3].

Checklists

Checklists are powerful repeatable repre-

Figure 1: *Types of Process Documents and Their Relationships*



Documentation Framework

Policies
"Laws" or "Principles" that govern operations.

Standards
Sections/Descriptions and "Operational Definitions."

Govern/Guide

Processes
"What happens over time" to produce results.

Implemented by

Procedures
"How-to" or step-by-step instructions.

Supported by

Training
Provides the needed knowledge and skills.

Tools
Supports and automates operations.

Note: Slide adapted from "A Software Process Famework for the SEI Capability Maturiy Model." Olson, et al. CMU/SEI-94-HB-01 sentations of activities that need to be completed to declare something completed. What makes checklists so powerful is that it usually does not matter in what order the checklist is completed. This is why checklists are useful for concurrent activities (e.g., versus flowcharts that are poor at representing concurrency).

Forms

Forms, along with instructions for completing the forms, are repeatable mechanisms for supporting processes. Forms are powerful mechanisms for collecting data in a repeatable way.

Step/Action Tables

One effective way to represent a procedure is using a step/action table [1]. Step/action tables are useful when order matters. For example, if a person needs to track his or her time, then starting to track time should not be the last step. For an example procedure of when order matters, see Table 5.

Some Success Stories

The best practices discussed in this article have been successfully used in practice over the last decade. More recently, major breakthroughs in defining extremely short and concise processes have also been achieved. An example of a short subprocess can be found in Figure 2 and Table 3. The following are some success story summaries (without revealing organizational identities).

Organizational Example 1

Organization No. 1 had a process that was not followed very well, and the process user feedback was that users did not like the process. The process had the following weaknesses (which are typical to most organizations):

- Mixture of document types: policies, standards, processes, and procedures.
- The existing processes lacked pictures (i.e., mostly text).
- The principle of *chunking* was violated in the flowcharts and in the number of process and procedure steps (e.g., processes and procedures with more than 20 steps), making the processes and procedures hard to use.
- Processes did not address all the five W's (e.g., *when* was missing; some other W's were also weak). It is hard to follow a process if you do not know when to start or when you are done.
- Procedures were very large and hard to follow (e.g., typical of International Organization of Standardization [ISO]

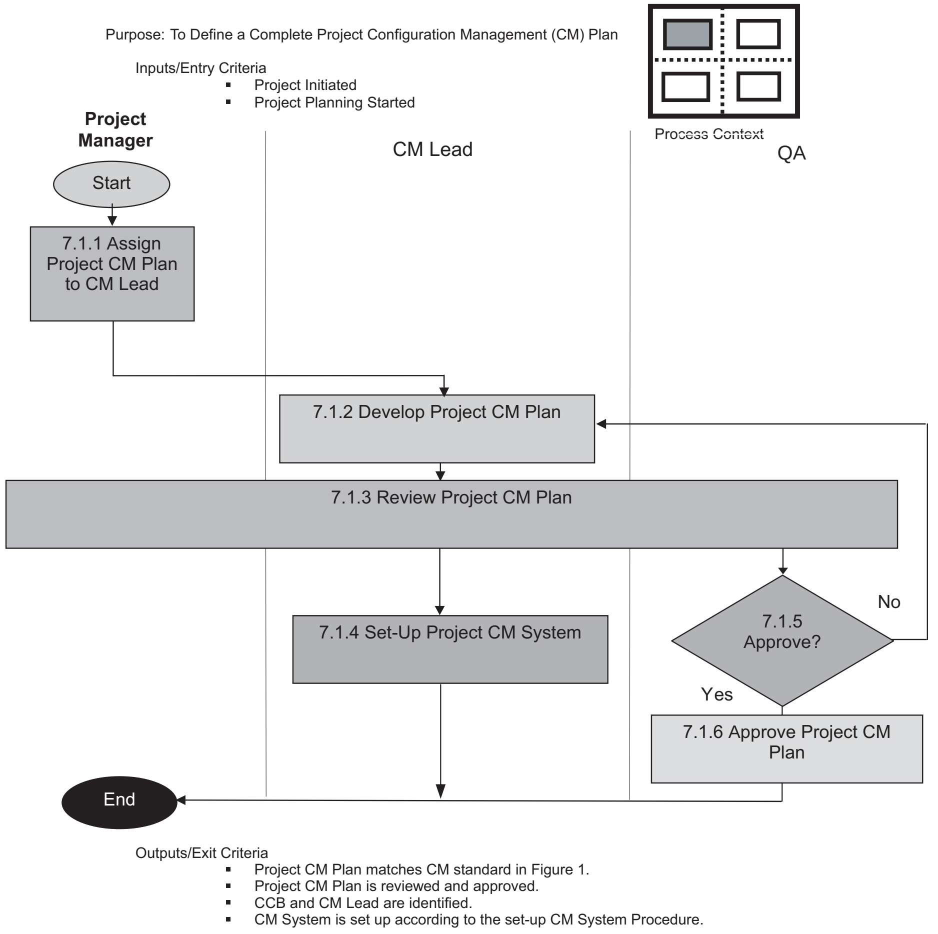

Purpose: To Define a Complete Project Configuration Management (CM) Plan
Inputs/Entry Criteria
Project Initiated
Project Planning Started
Process Context
Project Manager
CM Lead
QA
Start
7.1.1 Assign Project CM Plan to CM Lead
7.1.2 Develop Project CM Plan
7.1.3 Review Project CM Plan
7.1.4 Set-Up Project CM System
7.1.5 Approve?
No
Yes
7.1.6 Approve Project CM Plan
End
Outputs/Exit Criteria
Project CM Plan matches CM standard in Figure 1.
Project CM Plan is reviewed and approved.
CCB and CM Lead are identified.
CM System is set up according to the set-up CM System Procedure.

Figure 2: *7.1 Develop Project Configuration Management Plan (Expert Mode)*

Step	Role	Process Step
7.1.1	PM	**Assign Project CM Plan to CM Lead** At the appropriate time in the project (typically during project planning), the PM assigns the Project's CM plan development to the CM lead. **GUIDANCE:** A CM lead should have experience in setting up CM systems and performing CM. The CM Lead should also have been trained in CM.
7.1.2	CM Lead	**Develop Project CM Plan** The CM lead develops the Project CM plan according to the sections in the CM Plan Standard. It is required to follow the exact format of the CM Plan Standard **GUIDANCE:** The CM Plan Standard and example completed Project CM plans can be found on the Organizational Process Asset Web site.
7.1.3	All Active Roles	**Review Project CM Plan** The PM, CM Lead, and QA are required to peer review the CM Plan according to the CM Plan checklist in Appendix C. QA should review the CM plan against the CM Plan Standard. **GUIDANCE:** It is recommended to include the Project Team in the review.
7.1.4	CM Lead	**Set Up Project CM System** The CM Lead sets up the CM system for the project according to the Set-Up CM System Procedure in Appendix D. This step should be performed concurrently with steps 7.1.5 and 7.1.6. **GUIDANCE:** This task is only done once and may already be completed for maintenance projects.
7.1.5	QA	**Approve?** If "YES," then proceed to Process Step 7.1.6. If "NO," then procedd back to Process Step 7.1.2 for rework. After the third "No" iteration or 2 weeks after the first disapproval, escalate a CM Plan disapproval issue to Project Management. **GUIDANCE:** QA should ensure the CM Project Planning Process is followed. QA also needs to work with the Project Manager regarding approval issues.
7.1.6	QA	**Approve Project CM Plan** QA approves the Project CM Plan and places the plan on the project's official Web site. This sub-process is not complete until all steps have been completed (e.g., 7.1.4). **GUIDANCE:** QA should ensure the Project CM Plan is integrated into the overall Project Plan. The Project CM Plan is placed under CM with the Project Plan in the Project Planning Process.

Note: Project Manager (PM), Configuration Management (CM), Quality Assurance (QA).

Table 3: *7.1 Develop Project Configuration Management Plan (Intermediate Mode)*

procedures).

New processes and procedures implemented the best practices described in this article and addressed all these weaknesses above. Process modeling was used to add good diagrams and address chunking of the flow chart. The new processes addressed all the five W's and were defined on one page for expert mode (Please see Figure 2 for an example of the five W's). In conclusion, organization No. 1 was much happier with diagrams on one page (i.e., process models), and with short, usable processes and procedures.

Organizational Example 2

In organization No. 2, although detailed procedures existed, the overall processes had not been documented. Four sub-processes were defined on one-page diagrams (i.e., process models) for each process in expert mode (the five W's on one page), and a page of text along with guidance was developed in intermediate mode to support the diagram (see Figure 2 and Table 3 for a similar example). This organization packaged these four short processes in a single process guide in intermediate mode in about 20 pages total – four pages per process (similar processes documents can be more than 100 pages).

The experienced manager of this newly defined process was promoted in organization No. 2. Sometimes experienced people get stuck in positions because they are the only ones who know the processes, and often the processes are not documented. The new process document also helped with the transition because it allowed a new manager to come in and quickly learn the documented processes.

Organizational Example 3

Organization No. 3 requested a process review of all its processes that identified strengths, weaknesses, and made specific recommendations to senior management. During the organizational process review, it was discovered that a key process was not documented. The process was designed into three sub-processes, and each sub-process was defined on one page using a diagram (i.e., expert mode), and a page of text (along with guidance) was developed in intermediate mode to support the diagram (see Figure 2 and Table 3 for a similar sub-process example). The entire new process guide (including the policy, standard, process, and procedures) is about 20 pages long total (this includes about a dozen sections, including purpose, audience, usage, scope, metrics, procedures, references, etc). The new process also met all ISO requirements, which added about five pages. In conclusion, organization No. 3 is much happier and now has more complete, better, shorter, and more usable processes and procedures.

Key Process Question	Process Element
Why is the activity performed?	1. Purpose
What actions are performed?	2. Activities
What work products are used?	3. Input(s)
What work products are produced?	4. Output(s)
When does the activity begin?	5. Entry criteria
When does the activity end?	6. Exit criteria
Who performs activities?	7. Roles
Where is activity performed?	8. Process Context (e.g., Hierarchy)
How is the activity implemented?	9. Sub-Activity or Procedure

Table 4: *The Practical Process Questions*

Organizational Example 4

Organization No. 4 complained that its processes and procedures were too large and difficult to use. After applying the best practices described in this article, the processes and procedures were cut in half (e.g., 600 total pages were reduced to 300 total pages). The processes and procedures are also more usable. What is fascinating about this example is although the processes and procedures were cut in half, no information was lost.

Some Lessons Learned

Here are some of the lessons learned while defining processes with best practices:

- Do not mix policy, standard, process, and procedure information (e.g., in the same paragraph). Label this different information, and consider how the information is used.
- Define all process documentation as simply as possible, but no simpler (information that is too simple does not work). Keep process documentation concise (i.e., short and sweet), but expect some processes to be complex.
- Use good pictures (most people prefer pictures). Process modeling is a best practice and scales up to very complex systems. Use process modeling to develop good pictures.
- For each process or sub-process, define the five W's on one page using a diagram (see Figure 2 for an example). A good process diagram can replace 20-25 pages of text.
- Use procedures (i.e., checklists, forms, and step/action tables [1]) for implementing processes and for repeatability.
- Use chunking (i.e., seven plus or minus two), organize the chunks, and label the chunks (so users can find information quickly). Process modeling and information mapping [1] help

Table 5: *Example: Step/Action Table Procedure*

Step	Action
1	Begin to track time (e.g., write down the start time).
2	Look for defects in the selected work product by using the appropriate data-driven checklist.
3	Log the defects of the Defect Form. Continue logging defects until the work product is completely inspected using the checklist.
4	End tracking time (e.g., write down the end time). Calculate the total time spent looking for and logging defects, and record the total time on the Defect Form.

tremendously with this principle.

- Account for beginner, intermediate, and expert users of the process.
- Design measurement into the process. Do not add measurement as an afterthought.
- The processes must be tailored to each organization, each business unit or division, and each project.

Summary

In summary, the objectives of this article are the following:

1. Describe common problems with process documentation, including some human aspects of using process documents.
2. Discuss some best practices for defining short and usable processes and procedures.
3. Describe some success stories in real organizations.
4. Provide some lessons learned.

Defining short and usable processes and procedures is challenging. There are many best practices that can be used to help improve process documentation. The approach summarized in this article uses a collection of best practices, all wrapped into a process (for defining processes). The author hopes that the readers have benefited from the description of some of the best practices along with the example process in Figure 2 and Table 3.◆

Note

1. *Chunking* example: Most people can only remember *chunks* of information (e.g., that is why 15-16 digit credit cards numbers are broken into smaller chunks [seven, plus or minus two]; a 16-digit Visa number is usually broken into four groups of four digits).

References

1. Horn, Robert E. Mapping Hypertext: Analysis, Linkage, and Display Knowledge for the Next generation of On-Line Text and Graphics. Lexington, MA: The Lexington Institute, 1989.
2. Olson, Timothy G., et al. "A Software Process Framework for the SEI Capability Maturity Model." CMU/SEI-94-HB-01. Pittsburgh, PA: Software Engineering Institute, 1994.
3. Olson, Timothy G. "Defining Software Processes in Expert Mode." Software Engineering Process Group Conference, Atlanta, GA, 1998.
4. Ross, Douglas T., and Kenneth E. Schoman Jr. "Structured Analysis for Requirements Definition." IEEE Transactions on Software Engineering SE-3 (1) (1997): 6-15.

About the Author

Timothy G. Olson is founder and president of Quality Improvement Consultants, Inc. While performing quality consulting, Olson has helped organizations measurably improve quality and productivity, save millions of dollars in costs of poor quality, and reach higher Software Engineering Institute maturity levels. He has been formally trained in Crosby, Deming, Juran, International Organization of Standardization, Capability Maturity Model® (CMM®), CMM Integration^{SM}, and Six Sigma quality approaches. He is currently a senior member of the American Society of Quality and a member of the Institute of Electrical and Electronics Engineers. He has a master's degree from the University of Massachusetts.

Quality Improvement
Consultants, Inc.
7117 Obelisco CIR
San Diego, CA 92009
Phone: (760) 804-1406
Fax: (760) 804-1406
E-mail: tim.olson@qic-inc.com

Web Sites

Earned Schedule

www.earnedschedule.com

Earned Schedule (ES) introduces the concept of adding ES into a project management repertoire. ES discusses the difficulty of measuring schedule performance in dollars instead of units of time. ES is derived from Earned Value Management (EVM), but explores alternatives to EVM schedule tracking. It is derived from and is an extension to EVM. No additional data is needed for acquiring the ES measures; only the data from EVM is needed. In contrast to the cost-based indicators from EVM, the ES schedule performance indicators are time-based, making them easier to comprehend. The ES indicators provide a status and predictive ability for schedule, analogous to the facility for cost using EVM.

Software Technology Support Center

www.stsc.hill.af.mil

The U.S. Air Force's Software Technology Support Center (STSC) provides the Guidelines for Successful Acquisition and Management of Software Intensive Systems at <www.stsc.hill.af.mil/resources/tech_docs/gsam4.html>. A streamlined version of the content is provided and broken up into chapters for desktop usable access, as well as an overview of important software acquisition and development topics, helpful checklists for rapid self-inspection, and acquisition and management of software pointers. The STSC also oversees CrossTalk, offering monthly articles aimed at the software technology industry.

Software Engineering Institute

www.sei.cmu.edu

Carnegie Mellon University's Software Engineering Institute (SEI) has had the national mandate to advance the state of the practice of software engineering and to serve as a national resource in software engineering and technology. SEI aims to help organizations and individuals improve their software engineering management practices. SEI provides services dealing with the major categories of software engineering, including management, engineering and acquisitions. The Web site also provides links to education, training, and frequently asked questions in its mission to support software engineering professionals and advance software engineering and related disciplines to ensure the development and operation of systems with predictable and improved cost, schedule, and quality.

Should Your Projects' Leaders Be on Springer?

Paul Kimmerly
U.S. Marine Corps, Technology Services Organization

This issue of CROSSTALK *focuses on software projects that fail. Process improvement programs can help a software program succeed. They can also help it fail. Process improvement programs have the potential to be dysfunctional for their own reasons. To help put this in perspective, I chose to look at the ringmaster of dysfunction, Jerry Springer. This article draws a parallel between the problems faced by process improvement programs to the experiences related on* The Jerry Springer Show.

Process improvement programs face a number of challenges within an organization. These challenges can include fickle or uncaring sponsors, strange behavior, and process envy. Such dysfunctional behaviors reflect the personalities and culture of the organization. Any organization is a miniature version of society as a whole. In order to gain some insight into the effects of dysfunctional behavior in an organization, let us turn to the best demonstration of dysfunction in our society, *The Jerry Springer Show*. This article will relate some of the dysfunctional behavior seen on the *Springer* show that sabotages relationships to the resistance and strange behavior that can sink a process improvement program.

Sponsors

Many articles and presentations on successful improvement programs talk about the importance of sponsorship. Involved and active sponsorship is critical to process improvement efforts. If the sponsor gets involved, others in the organization notice and play along. There are two types of sponsor behaviors that can hurt an improvement effort: cheating and a lack of attention.

The Springer show routinely features cheating spouses. Invariably, they bring their new lover on the show and a fight ensues. Feelings are hurt and the relationship is never the same. In the process improvement world, cheating spouses relate to sponsors that jump from improvement idea to improvement idea. Reading too many airline magazines often causes this problem. A sponsor will profess his love for Capability Maturity Model Integration^{SM} (CMMI®)-based process improvement one day, staff an improvement program, and ask his managers to join in. Then, while the sponsor is on a business trip, a new improvement program catches his eye. Maybe it is Lean Six Sigma. Maybe it is the Balanced Scorecard. Suddenly that CMMI book just does not look the same. Its cover isn't as shiny and his attention wanders. The rest of the organization will notice this and follow his lead. This can lead an organization down a long road of shifting from one improvement effort to another, and it does not take long for the organization to see improvement efforts as shallow and pointless. It takes the full commitment of the sponsor to win the hearts and minds of the organization.

> *"The sponsor pays close attention while the process is growing up, but can easily become content once the target grade level is achieved. The sponsor may forget that the improvements came about because of their involvement."*

Even if sponsors remain committed to a single improvement program, they can lose focus. If that happens, subordinate managers who never really bought into the improvement program will see an opportunity to resist changing their behavior. Organizations that focus on the grade for a CMMI appraisal are especially susceptible to this behavior.

Another regular event on the *Springer* show involves parents who are suddenly confronted with the unexpected and wild behavior of their children. They say things like, "He's always been such a good child." Then the child comes out dressed as a vampire or wearing a diaper and brings out his or her equally bizarre friends. At one time, these were probably good, attentive parents. However, as the children grew older and started hanging out with their friends and pursuing other interests, the parents felt their job was done and turned to other interests of their own.

This type of inattentive parenting equates to sponsors and managers who rush to put improvements in place to get the grade. The sponsor pays close attention while the process is growing up, but can easily become content once the target grade level is achieved. The sponsor may forget that the improvements came about because of their involvement. Once the grade is achieved, the sponsor may turn attention to more pressing matters with the expectation that things will not change. Project managers not committed to continuing improvement will begin to stop following the processes they do not like or do not see as important. While the sponsor is saying, "It has always been a good project. It's Level 3, you know," things suddenly start to change. The project starts to miss deadlines, and status reports lose their clarity and validity. Customers start complaining and the sponsor cannot understand what went wrong. Sponsors need to continue to stress the importance of good processes and support their Quality Assurance (QA) group, which can often be the first source of information when a project's behavior turns strange. A well-established and supported QA group can be a surrogate parent to make sure the projects continue to follow their processes as intended.

Project Managers

Project managers play an integral role in continuing improvement efforts. Successful improvements require cascading sponsorship that flows from the sponsor through the higher levels of management down to the practitioners. Unfortunately, many mid-level managers resist improvement and the changes associated with it.

On the *Springer* show, many of the conflicts stem from the refusal of one person to commit to a long-term relationship. Promises of marriage never result in a trip to the altar. Eventually, one of the people in the relationship strays. Then, they show up on *Springer* with a new love (or two) and a fight ensues. Managers who resist often say the right things in the presence of the sponsor, but never fully commit to making improvements. Like the proposal that never comes, the improvements are constantly delayed. Some small change may be made when sponsors are looking, but the managers never commit when left on their own. If the sponsor does not press the issue, the desired change will not happen.

Cross-dressers often make appearances on the *Springer* show. These people present one face to the public while behaving very differently behind closed doors. The *Springer* show lets them expose their true selves on national television. Projects often behave the same way. It is not unusual for a project to develop a lot of process documentation and store it on their organization's repository for all to see. They might even make a big show of producing the documentation. The public face looks the way it should; however, when a closer look is taken, the documentation is all for show. The project continues to behave the way it always did without making the changes that were documented. This often happens when a project tries to write documentation that matches the way it *thinks* things should be done, not the way they *are* done. Most organizations have some level of informal process in place to produce software. It may not be written down, but people know the general way that things are done. An easy first step in an improvement program is to document the current process and then look at where improvements are needed. Projects that want to look like they are doing the right thing often take a different approach. They give someone the task to go off in a corner and write the documentation without involving the people performing the work. When that happens, the documentation looks great. Unfortunately, no one follows it because they have no stake in what it says.

Resistance to change is to be expected in life and in process improvement. Some resistance is more obvious. When people resist the changes in their relationship, chairs fly across the stage at the *Springer* show. In organizations, some projects are just as open in fighting change. Compared to hidden resistance, this kind of resistance is easier to handle because it is so obvious. On *Springer*, Steve and the security crew know to rush the stage and get between the combatants. In an organization, the sponsor and the process improvement group need to know where to step in and take action.

Sometimes, *Springer* reveals secret crushes where one guest longs for another, but fears to come out in the open. This can actually be a good situation for an organization. If a sponsor praises a project with proven, successful processes, other projects will want similar attention. They will secretly crave the sponsor's attention. Such feelings can be leveraged to bring improvements to those projects.

Improvement Groups and Appraisals

Steve and his security staff sit right at the edge of the *Springer* stage ready to jump in and get in the middle of any problems that arise. Process improvement groups serve the same role in an organization. When unexpected conflict or resistance appears, the process group is there to step in and work with all the involved parties to reach a solution. Sometimes the combatants on *Springer* do not want to stop, just like projects that always resist. Steve and his crew keep coming back until the guests behave. Process improvement groups and sponsors need that same level of commitment and persistence.

At the end of every show, Jerry sits off to the side and provides a little homily. He summarizes what the audience saw and puts it all into perspective as much as possible considering what is usually on the show. Lead appraisers fill that same role in improvement efforts. What they see in an organization may be as ugly as what Jerry deals with every day, but they are supposed to stay above it all and put the organization's behavior into perspective. That is, of course, unless the lead appraiser has been working as a paid consultant with the organization on its improvement efforts. That relationship can be a little *too* close, like some of the family members that show up on the *Springer* show. Such behavior from consultants may produce misleading or questionable appraisal results.

Summary

Through it all, Jerry and his security staff see the problems of society pass in front of them every day. They see all manner of people from society's mainstream to its fringes. Now, this article is not intended to encourage process improvement groups to watch *The Jerry Springer Show* for guidance or insight, even though all of the behaviors listed here can probably be seen in a one-hour episode. Watching the show is its own form of dysfunctional behavior.

In an organization, a process improvement group is likely to get a *Springer*-esque view of behavior from dealing with all levels of the organization. Dysfunctional behavior can come from the sponsor, the middle managers, or even the practitioners. Whatever the source or nature of the behavior, it can cause a process improvement program to fail. Failed improvement programs can have a larger affect on the organization as a whole. The *Springer* show puts extreme dysfunctional behavior on the public stage where we can all see its effects. Improvement groups and appraisals can do the same for an organization by showing the dysfunctional process behavior to the sponsor and working to address it.◆

About the Author

Paul Kimmerly has 17 years experience in software development for the different incarnations of the U.S. Marine Corps Technology Services Organization in Kansas City. A member of the Software Engineering Process Group (SEPG) since 1993, Kimmerly has served as the group's chair for the past nine years. Paul is an authorized Standard CMMI Assessment Method for Process Improvement Lead Appraiser. He presented at the 1997 and 2000 Software Engineering Symposiums and the 2004 National SEPG Conference. Kimmerly has contributed several articles on process improvement to CrossTalk magazine.

DFAS-KC/TKGB
1500 E 95 ST
Kansas City, MO 64197
Phone: (816) 926-5364
DSN 465-5364
Fax: (816) 926-6969
DSN 465-6969
E-mail: paul.j.kimmerly@dfas.mil

When Failure *IS* an Option …

I'm sitting here writing this BackTalk while on business in Baltimore. I flew into the Baltimore Washington International Airport (BWI). I remember a few years ago when BWI advertised itself as a great alternative to both Reagan National (in downtown D.C., always crowded) and Dulles (which is about 25 miles out of D.C.). BWI was convenient to both Baltimore and D.C., and small enough that rental cars were located within a five minute walk of the terminal. Things have changed! BWI is now under construction, and I had to walk from one end of the airport to the other to get to baggage claim. Then, I had to walk all the way to the other end of the airport to catch the "rental car bus." Car rentals are now about five miles away, so you have no option other than the inconveniently located rental car bus – and the buses were extremely crowded. Things change. What used to be a *good thing* becomes inconvenient and appears to be poorly designed.

Which brings us to "Why Software Fails." This has been a hard column to write – I was tempted to take the easy way out, and simply list and add the pictures of a few former co-workers and acquaintances who, in my opinion, have contributed to failing software over the years.[1] But instead, I have come up with a good start at a list that explains why software fails.

Software fails … one day at a time. Insidious little events occur. Small errors creep in. You have the *road not taken* syndrome. You realize that you could do better, but you don't have time to go back and start all over again. It's not always the big errors that cause failure, it's the little errors that accumulate.

Software fails … with the best of intentions. Developers, with the exception of a few TRULY unspectacular folks I have known, don't really set out to do a poor job.[2] We try and make the right choices, but we don't have the ability to predict the future. If things had turned out a little differently, we would have had a spectacular success. Instead, decisions turned out to be sub-optimal. If we only had the time to do it over.

Software fails … because we have no other choice. Sometimes politics, budgets, and schedules force us to make decisions we don't like. In a perfect world, we would have the time and budget to make perfect software. The world isn't perfect, and we are often forced into less than perfect solutions. We know better, we just can't do better. Real-world requirements change and we have to make the software react also, or the software becomes obsolete. I am relatively sure that the designers of the new BWI rental car terminal wish they were still located within a few hundred feet of the airport. However, the reality is that increased air traffic and congestion made this option infeasible. Sad to see it go, but it beats NOT having a rental car, doesn't it?

Software fails ... because we are overcome by events. Sometimes, we have to make choices before we have time to research all of the options. Schedules are tight and it's more important to make a workable decision now rather than making a better decision later. We don't like it, but it's just what we have to do.

Software fails … because we can't think of everything. Ever left the house for the grocery store with a *memorized* grocery list? You started out for milk and eggs. You added carrots and sliced cheese. Your spouse reminded you that you need toothpaste and shampoo. Only a few items. Yet, by the time you get to the grocery store, you're reduced to calling home, because all you can remember is milk, eggs, and *something else*. How many things can you juggle in your memory at one time? For most people, I suspect this number peaks out around nine or 10. Unfortunately, large-scale software has millions of lines of code, and literally tens of thousands of function points. How can we comprehend such large scale? We use architectural design to decompose the problem, and we use high-level languages, modularity, and object-oriented techniques to further break the problem down. However, with software of such large size, things just slip through the cracks. Requirements are missed or not implemented. Obvious errors are usually obvious only in hindsight – after the failure.

Software fails … because developers are only human. Have you ever spent hours (or even days) looking for an error, and had somebody wander by, glance over your code, and immediately see the problem? When you develop code, you tend to internalize your own errors, and then your brain fails to see them. An outside observer, however, can often see what you keep overlooking. Almost all good developers know that you need somebody else to review your work. This applies to all phases of software development: requirements, design, coding, and maintenance. One of my favorite quotes is, "When quality is vital, independent checks are necessary, not because people are untrustworthy but because they are human[3]." Even if you are one of the best software developers around,[4] you make mistakes. So does everybody else.

Software fails … because you failed to consult the Software Technology Support Center (STSC) for help when developing your software. Or, if not the STSC, you should learn from *somebody*! You want to emulate the best practices of others while at the same time keep from making the same mistakes that others have made. Learn from the mistakes of others and also learn from the success of others. Find out what other similar development efforts did right and wrong. Read journals. Talk to fellow developers on other projects. But then – you *are* already reading CrossTalk, aren't you?

— David A. Cook, Ph.D.
The AEgis Technologies Group, Inc.
dcook@aegistg.com

P.S. I am not claiming my list is complete or even valid (after all, I didn't review this with anybody else!). Feel free to e-mail me your additions or comments, and maybe you'll see them in a future BackTalk column.

Notes

1. Worried that you'll find your name here, aren't you?
2. STILL worried that you'll find your name here, aren't you?
3. Humphrey, Watts S. Managing the Software Process. Addison-Wesley, 1989.
4. Well, you certainly aren't expecting to find any name other than mine, are you?

CrossTalk is co-sponsored by the following organizations:

76th Software Maintenance Group

309th Software Maintenance Group

402d Software Maintenance Group

U.S. Department of Homeland Security

Homeland Security

NAVAIR

CrossTalk/517 SMXS/MDEA
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820